SCIENCE CHINA
Information Sciences

• RESEARCH PAPERS •

May 2010 Vol. 53 No. 5: 964–979
doi: 10.1007/s11432-010-0096-7

Secure, dependable and publicly verifiable distributed data storage in unattended wireless sensor networks

REN Wei[1]*, REN Yi[2] & ZHANG Hui[3]

[1]*School of Computer Science, China University of Geosciences (Wuhan), Wuhan* 430074*, China;*
[2]*Department of Information and Communication Technology, University of Agder (UiA), Norway;*
[3]*Department of Electronics, Politecnico di Torino (POLITO), Italy*

Received July 13, 2009; accepted February 3, 2010; published online April 7, 2010

Abstract In unattended wireless sensor networks (UWSNs), sensed data are stored locally or at designated nodes and further accessed by authorized collectors on demand. This paradigm is motivated by certain scenarios where historical or digest data (e.g., average temperature in a day), instead of real-time data, are of interest. The data are not instantly forwarded to a central sink upon sensing, thereby saving communication energy for transmission. Such a paradigm can also improve data survivability by making use of distributed data storage in cheap flash memory on nodes. However, the security and dependability of such data storage are critical for the future data accessibility in UWSNs. To address this issue, we propose a secure and dependable distributed storage scheme for UWSNs. Our scheme takes advantages of both secret sharing and Reed-Solomon code, which can achieve computational security and maintain low communication overhead in terms of shortened data dispersing size. We also propose a general coding method to publicly verify data integrity in a distributed manner, with low communication and storage overhead, and without the need of holding original data. The extensive analysis justifies that our scheme can provide secure, dependable and publicly verifiable distributed data storage in UWSNs even in the presence of node compromise and Byzantine failure.

Keywords secure storage, dependability, data integrity, wireless sensor networks, wireless network security

Citation Ren W, Ren Y, Zhang H. Secure, dependable and publicly verifiable distributed data storage in unattended wireless sensor networks. Sci China Inf Sci, 2010, 53: 964–979, doi: 10.1007/s11432-010-0096-7

1 Introduction

Wireless sensor networks (WSNs) consist of a large number of sensor nodes that can be easily deployed to various environments. The sensor nodes usually generate a large amount of data over their lifetime to report or record the environmental context. Data storage in WSNs mainly falls into two categories, namely centralized data storage and distributed data storage [1]. In the former case, data are sensed, processed, aggregated and managed at a central location, usually a sink. In the latter case, after a sensor node has generated some data, the node stores the data locally or at some designated nodes within the network, instead of immediately forwarding the data to a centralized location out of the network [2–4]. Such a paradigm is called "Unattended WSNs" (UWSNs) [2].

*Corresponding author (email: weirencs@cug.edu.cn)

UWSNs paradigm is motivated by scenarios whereby not real-time information, but digest information is of interest. The digest information includes historical summarization or locally extracted results on data processing, such as the average temperature during last three months; the highest and lowest humidity degree during last 24 h; or more specifically, the average concentration of a chemical element in soil during last half year [5]. The digest information is provided on demand upon user's retrieval, avoiding frequent data transmission to sinks. It thus remarkably saves the communication energy. Moreover, there exist some scenarios that may not be convenient for deploying robust real-time sinks, e.g. in the sea, volcano, or places far away from collectors. The data have to be stored locally or in designated nodes in the network and wait for further collection.

Security and dependability of the stored data in UWSNs are of paramount importance for their functionality. The reasons are listed as follows: 1) As sensor nodes are exposed to unattended environment, data may be lost due to failure of sensor nodes, such as power depletion, melting, corroding or getting smashed; 2) data may be stolen or modified by mobile adversaries [6] that can compromise a selective subset of sensor nodes. Hence, a fault-tolerant and compromise-resilient scheme for distributed data storage has to be in position to ensure data availability, integrity and confidentiality.

The straightforward data redundancy or fault-tolerant methods for data availability cannot be applied in UWSNs domain that has resource constraints. Besides, a naive way to check the data integrity of distributed data by hash value is also not appropriate. More precisely, the hash values of the distributed data are stored at an integrity checker or verifier, and the checker sends requests to data holders to launch a check. If the computed hash values of returned data do not match stored hash values, the data are modified, thus obviously incurring high communication overhead due to the overwhelming data communications. It also induces remarkable storage overhead as the checker has to store a large volume of hash values. Especially, such a checking method is not reliable as the checker is a single point of failure. Hence, a lightweight method for data integrity check with distributed public verification is mandatory.

Currently, UWSNs paradigm, especially its security issues attract more and more attentions in research communities [2–5, 7, 8]. Nevertheless, security, dependability, and integrity verifiability of distributed data storage still require thorough exploration and tailored design.

Objective of the paper: The goal of this paper is to investigate techniques for achieving the secure and dependable distributed data storage as well as lightweight and public verifiability of integrity for UWSNs.

Contributions: 1) In this paper, we propose a secure and dependable data distributed storage scheme by making use of secret sharing and Reed-Solomon code. Such a scheme is resilient for random fault and node compromise, with communication and storage efficiency maintained. 2) We present a generic technique called two granularity linear code (TGLC), which can verify distributed data's integrity publicly and efficiently, i.e., in a distributed manner and with low communication and storage overhead without original data.

The remainder of the paper is organized as follows: The network assumption, adversary model and design goals are presented in section 2. Section 3 presents some preliminaries on secret sharing and error correcting code. Section 4 proposes the data storage scheme. The extensive performance analyzes and security analysis are given in section 5. Section 6 proposes a lightweight coding method for public verifiability of data integrity. Finally, section 7 concludes the paper.

2 Problem formulation

2.1 Network assumptions

We consider a sensor network that is composed of a large number of sensor nodes with unique ID, and many users or collectors can access the data in the network on demand. Following the previous works on distributed storage in WSNs (e.g. Unattended WSNs [2, 7], Storage-Centric WSNs [3], Data Centric WSNs [9], In-Situ Data Storage WSNs [5], etc.), we assume that each sensor node has certain capacity to store data. (The Gigabyte memory for sensor node is already available [10, 11].) Once the sensing

data are generated at sensor nodes, they will be stored locally for future retrieval. The stored data will be accessed by authorized users, instead of sensor nodes, e.g., the soldiers with laptop in a battlefield. The collected data may also be further processed together at in-door workstations or powerful hand held devices that have no computational constraints. We assume that the pairwise shared key between any two nodes can be established easily, e.g., by employing the key establishment scheme [12].

2.2 Attack model

The UWSNs impose some attacks in many ways. In this paper, we focus on the attackers who attempt to eavesdrop, physically corrupt, modify, and capture the stored data in networks. The attackers thus have the following capabilities:

(1) The attackers can eavesdrop on all the traffics in UWSNs.

(2) The attackers may randomly select some sensor nodes to physically corrupt them (such as smash, melt or corrode), or sensor nodes fail due to power depletion. In this occasion, the nodes totally lose the functionality.

(3) The attackers can compromise some sensor nodes, and then modify the data stored in the nodes. This type of attacker can further be categorized into two classes: 1) The adversary compromises the node to pry or modify some data and then leave. 2) The adversary compromises the node and comes back periodically or occasionally for some purposes. The left type is the most advanced adversary who resides at compromised nodes for a long term, which acts normally in data storage stage but malfunctions at data submission stage. They can be easily detected upon data retrieval and thus are out of the scope of this paper.

2.3 Design goal

Our design goal is to provide the security and dependability of data storage in the presence of aforementioned attacks. We focus on the security in terms of data confidentiality and integrity, and the dependability in the sense of the tolerance of random fault and the resilience of security compromise. That is, if a random fault occurs, the data still can be recovered by the user upon request, and data confidentiality and integrity are retained. Also, if some nodes are compromised, the data modification is still detectable and data secrecy is still protected. Meanwhile, to defend the eavesdropping attack, data have to be encrypted in transmission.

2.4 Evaluation metrics

The basic security requirements are satisfied as follows: 1) Data confidentiality. Data are encrypted and only authenticated users can access it. 2) Data integrity. If data are modified by attackers, they can be detected by users. 3) Data availability. The stored in-networking sensed data can be retrieved correctly upon request.

The performance evaluation includes the storage, communication, and computation consumption.

The dependability is evaluated in two folders: 1) Compromise resilience. Provided that some nodes are compromised by attackers, data integrity and confidentiality can be achieved. 2) Fault tolerance. In the case of the random faults in the data, the original data can be recovered by the authenticated users.

The energy consuming has three parts: communication, storage and computation. Communication costs much more energy than storage. For example, transmitting data over radio channel consume 200 times more energy than storing the same amount of data locally on a sensor node [13]. Radio reception costs 500 times more energy than reading the same amount of data from local storage [10]. Computation cost depends on the calculation involved. Simple arithmetic operations such as addition or multiplication may have relatively low cost due to the powerful processor in the new generation sensor node [14, 15]. Therefore, we pay more attention to the communication cost.

3 Preliminary

3.1 Secret sharing

Shamir [16] proposed an (m, n) secret sharing scheme based on polynomial interpolation, in which m of n shares are required to reconstruct the secret.

Shamir's secret sharing: The secret k is in Z_p (p is prime, and $p > n$). Each shareholder i is in the set P ($|P| = n$). All mathematical operations are in the Finite Field Z_p. To distribute k, select a polynomial $a(x)$ with degree $m-1$ and constant term k. Generate a share s_i for each i in P with $a(x)$: $s_i = k + \sum_{j=1}^{m-1} a_j i^j$. s_i is also in Z_p. To reconstruct k, retrieve m coordinate pairs (i, s_i) of all i in authorized subset B of P ($|B| = m$) and use the pairs in the Lagrange interpolation formula: $k = \sum_{i \in B} b_i s_i$, where $b_i = \prod_{j \in B, j \neq i} \frac{j}{j-i}$.

3.2 Reed-Solomon codes

A (k, n) erasure code encodes a block of data into n fragments, which has $1/k$ size of the original block, so that any k can be used to reconstruct the original block. An example of such erasure coding schemes is Reed-Solomon codes [17].

Reed-Solomon codes: For $q \geqslant n$, let $\alpha_1, \ldots, \alpha_n$ be n distinct elements of Galois Field GF_q. The Reed-Solomon codes of message length k, with parameters $\alpha_1, \ldots, \alpha_n$ is defined as follows: Associate with a message $\boldsymbol{m} = \langle m_0, \ldots, m_{k-1} \rangle$ a polynomial $M(x) = \sum_{j=0}^{k-1} m_j x^j$. The encoding of $\boldsymbol{m}$ is the evaluation of M at the n given points, i.e., $Encode(\boldsymbol{m}) = \langle M(\alpha_1), \ldots, M(\alpha_n) \rangle$. By construction, the Reed-Solomon codes have message length k and block length n. The message can be reconstructed from any k blocks, and the codes can correct up to $t = \lfloor \frac{n-k+1}{2} \rfloor$ errors.

4 Proposed data storage scheme

In this section, we propose a family of schemes for secure and dependable data storage.

4.1 Notation

Suppose there are x sensor nodes in UWSNs. They are formulated as an undirected graph $G(V, E)$. The node set is $V = \{v_1, v_2, \ldots, v_x\}$ and the edge set is $E = \{e_1, e_2, \ldots, e_y\}$. The node is denoted by v_i ($1 \leqslant i \leqslant x$). The node v_i has d_i neighbors, which compose a set SET_i. The distributed data may be called *Shares*, to distinguish it from the original sensed data.

We list some major notations used in following sections in Table 1.

4.2 Basic setting

After sensor node v_i senses data $Data_i$, the listed steps begin to protect the data integrity and data confidentiality if the data are stored individually and locally:

(1) Generate a random session key RSK.

(2) Compute the keyed hash value of $Data_i$ using key RSK. The hash value, denoted by $MAC = h(Data_i \| RSK)$, protects data integrity.

(3) Encrypt $\{Data_i \| MAC\}$ using key RSK. The encrypted data, denoted by $\{Data_i \| MAC\}_{RSK}$, protect data confidentiality.

(4) Encrypt random session key RSK using symmetric key between sensor v_i and user u, or public key of a user u. Both are possible, depending on the underlying key management scheme, which is independent of our work. Because the public key does not concern the node compromising, and public key system is already possible in sensor networks, we assume the public key is used hereby, which is denoted by PK_u. We denote the encryption of random session key RSK by $\{RSK\}_{PK_u}$.

Therefore, the sensed data are equipped with the protection of data integrity and data confidentiality. $Data_i$ is equipped into $DTa_i = \{\{Data_i \| MAC\}_{RSK} \| \{RSK\}_{PK_u}\}$. Only authorized users can decrypt

Table 1 Notations

v_i	sensor node with ID i
SET_i	node i's neighbors
$Data_i$	data sensed by node v_i
$K_{i,j}$	pairwise secret key between v_i and v_j
PK_u	collector(user)'s public key
RSK	random session key
S_i	the distributed data shares with ID i

out RSK using corresponding private key upon receiving DTa_i, and then decrypt out the original data, and check data integrity through hash value.

4.3 Replication based scheme (RepS)

It is known that to store the data individually is not dependable. If the node is compromised or physically attacked, all data in the node will be lost. One way to improve the dependability is thus to replicate the data and distribute the replicas to the neighbors. If some nodes are compromised or physically attacked, the data can still be retrieved from the other nodes that store the replica. We thus present a replication based scheme (RepS) for distributed data storage as a baseline.

Scheme: v_i randomly selects n nodes in SET_i. For each node (e.g. v_j), v_i distributes one replica of data $S_t = DTa_i$. To differentiate the data, the packet includes the v_i and *seqno*.

$$v_i \rightarrow v_j : \{v_i \| seqno \| S_t\}, t \in [1, n], t \in Z.$$

The user collects data DTa_i from any one of n nodes that store the replica, decrypts it, and verifies the data integrity.

Analysis: Compared with individual storage, RepS does not improve the security but improve the dependability on fault tolerance. However, it induces much communication and storage overhead. More preciously, for n-replica scheme, the communication and storage overhead are both $|DTa_i| * n$.

4.4 Secret sharing based scheme (SSBS)

We observe that performance and dependability of RepS scheme can be further improved. For example, it is desirable to avoid the public key encryption operations in RepS (and in individual storage). Moreover, RepS is not dependable if a user is compromised or corresponding private key is exposed, attackers who capture only one replica can reveal the original sensed data. In contrast, secret sharing scheme can encode the sensed data into many shares to protect the confidentiality without inducing any encryption operation. Moreover, even if one user is compromised, the sensed data still cannot be revealed if the number of captured shares is less than a threshold value. Hence, to further improve the dependability of data storage with compromise resilience, we propose SSBS scheme (secret sharing based distributed data storage).

Scheme:

(1) v_i employs (m, n) secret sharing scheme to encode data $DTb_i = \{Data_i \| MAC\}_{RSK} \| RSK\}$ into n shares, denoted by $S_1, \ldots, S_n$;

(2) v_i randomly selects $n, (n < |SET_i|)$ neighbors in SET_i, e.g. $v_j \in SET_i$, and distributes one randomly selected distinct share to v_j by using pairwise secret key $K_{i,j}$ to encrypt the packet.

$$v_i \rightarrow v_j : \{v_i \| seqno \| S_t\}_{K_{i,j}}, t \in [1, n], t \in Z.$$

The user collects m shares in the nodes, reconstructs the DTb_i using Lagrange interpolation, decrypts it and checks data integrity.

Example: Suppose DTb_i has 2 bytes and uses $(6, 8)$ secret sharing scheme to generate 8 shares. Suppose field Z_{65537} is selected $(65537 > 2^{16})$. Select random polynomial $a(x)$ with degree 5 and constant term DTb_i. Generate a share S_i for each $i \in [1, 8], i \in Z$. $S_i = DTb_i + a_1 i + a_2 i^2 + \cdots + a_5 i^5$. All operations

are in field Z_{65537}. The distributed shares are S_i. Thus, each 6 in 8 shares can recover the DTb_i by Lagrange interpolation.

Analysis: SSBS improves the performance and dependability of RepS due to the properties of secret sharing. Secret sharing can provide perfect confidentiality of the secret in the sense of information-theoretical security. For (m, n) secret sharing, any shares less than m have completely uncertainty about the secret (i.e., DTb_i). Therefore, even if more than one node is compromised, the attacker still cannot reveal the data without enough shares.

In scheme SSBS, the length of distributed data shares is always shorter than that in RepS, as RSK is always shorter than $\{RSK\}_{PK_u}$. However, it confronts the same undesirable storage and communication overhead because in both RepS and SSBS, the distributed share size equals original data size (namely DTa_i or DTb_i). Moreover, the decreased computational overhead is public key encryption. The additional overhead is computation overhead for secrete sharing, which has $O(n*m)$ modular exponential operations, and data encryption for transmission. The computational cost of the data reconstruction from shares is omitted since the authorized users can reconstruct the data at workstation which is supposed to have no computational constraints, as pointed out in network assumption section.

Lou [18] proposed a multi-path routing based on threshold secret sharing scheme to provide data confidentiality without encryption for mobile ad hoc networks. It uses the same technique as SSBS for confidentiality, but later we will show that such a scheme can be further improved by inducing Reed-Solomon code.

4.5 Advanced hybrid scheme (HybridS)

To further improve storage and communication efficiency of SSBS scheme without loss of security and dependability, we notice that erasure code can reduce space and bandwidth overheads of redundancy in fault-tolerant data storage [19, 20]. We thus propose HybridS — a scheme based on secret sharing and erasure coding, which takes the advantages of both erasure code and secret sharing.

Parameter setup: Select n neighbors to distribute shares. To reconstruct data, m shares are required. Let $\lfloor \gamma * n \rfloor = m$, where γ is a security parameter (e.g., $0.5 \sim 0.8$). If γ is larger, a higher percentage of shares is required for the reconstruction of the data. Recall that $DTb_i = \{\{Data_i \| MAC\}_{RSK} \| RSK\}$, denoted by $\{E \| RSK\}$, where $E = \{\{Data_i \| MAC\}_{RSK}\}$.

Share generation:

(1) v_i employs (m, n) secret sharing scheme to encode RSK into n shares, denoted by $\underline{S_1}, \ldots, \underline{S_n}$.

(2) v_i employs (m, n) Reed-Solomon coding scheme to encode E into n shares. We denote the shares by $\overline{S}_1, \ldots, \overline{S}_n$. More specifically, v_i divides E into m parts, and selects

$$M(x) = E_0 + \cdots + E_{m-1} x^{m-1}.$$

Let $x = \alpha, \alpha^2, \ldots, \alpha^n$, ($\alpha$ is the primitive element in $GF(2^q)$, $n \leqslant 2^q - 1$). Let $\overline{S}_t = M(\alpha^t)$, ($t = 1, 2, \ldots, n$).

Share distribution: v_i randomly selects n neighbors in SET_i (e.g. v_j), and distributes one randomly selected distinct share $\underline{S_t}$ and $\overline{S}_t, t \in [1, n]$ to v_j by using pairwise secret key $K_{i,j}$ to encrypt the packet.

$$v_i \to v_j : \{v_i \| seqno \| \underline{S_t} \| \overline{S}_t\}_{K_{i,j}}, (t = 1, \ldots, n).$$

Data reconstruction: The user collects m shares in nodes and reconstructs RSK and E, and then reconstructs $\{DTb_i\}$ using Lagrange interpolation, decrypts the data, and checks data integrity.

Example: We give a trivial example to explain the main idea in above procedures. Node v_i distributes $DTb_i = \{E \| RSK\}$ using (6,8) scheme. Suppose E has 12 bytes and RSK has 2 bytes. Use (6,8) secret sharing scheme to share RSK to generate shares $\underline{S_i}$ ($1 \leqslant i \leqslant 8$). Use (6,8) Reed-Solomon code to encode E. More specifically, v_i divides E into 6 pieces: $E = \{E_0 \| \cdots \| E_5\}$. Each piece thus has 2 bytes. Let $Z_q = Z_{65537}$ and $M(x) = E_0 + E_1 x + ... + E_5 x^5$. Let $x = 1, \ldots, 8$ and calculate $M(x)$. All operations are in field Z_q. The shares are $\overline{S}_i = M(i)$ ($1 \leqslant i \leqslant 8$). Select a share from $\underline{S_i}$ and a share from $\overline{S}_i$ as a share unit. Distribute unit shares to 8 neighbor nodes. In the end, each 6 in 8 shares can recover the data E and RSK.

Analysis: The communication and storage overhead of HybridS is much lower than SSBS. More preciously, in (m, n) enhanced secret sharing scheme, the storage and communication overhead are $O(n * (\frac{|E|}{m} + |RSK|))$, where $|E| + |RSK| = |DTb_i|$. The total computational complexity for coding is roughly 2 times those of SSBS. Besides, the security and dependability of HybridS are as same as those of SSBS.

Discussion:

1) To further improve the security and dependability, the distributed data can be generated with different security parameters by using heuristics information, e.g., the energy status of the sensing node and the importance of the sensing data.

2) The security of RSK can be further improved by using multi-level secret sharing method [21], which distinguishes the importances of different shares with same share length for RSK reconstruction. For example, more important shares are distributed to safer nodes.

3) As many data retrieval schemes propose aggregative data collection, our scheme is also able to be compatible in this situation by replacing the naive secrete sharing scheme to homomorphic secret sharing scheme [22].

5 Analysis of secure and dependable storage

In this section, we analyze the performance, security and dependability properties of the proposed RepS, SSBS, and HybridS schemes. As the attack model is defined earlier, our security analysis will focus on the data that are modified or destroyed by random fault or mobile adversaries.

5.1 Performance analysis

We summarize comparisons between RepS (n-Replication), SSBS ((m, n) secret sharing) and HybridS ((m, n) enhanced secret sharing) in Table 2. Table 3 shows quantitative performance analysis in terms of storage, communication and computation overhead. As justified in section 2.4, the communication cost is most important. We can draw the conclusion that HybridS attains the best performance owing to the communication overhead (more specifically, the length of shares) and achieves the highest level of security and dependability.

5.2 Security analysis

Proposition 1. Even if adversaries compromise data holders and resides afterwards to corrupt the data (i.e., the most advanced adversary), the sensed data can still be able to reconstruct original data upon the first request with the probability $\frac{(n(1-p_m))!(n-m)!}{n!(n(1-p_m)-m)!}$, assuming $n(1-p_m) \geqslant m$, where p_m is the percentage of compromised nodes.

Proof. If the data are retrieved from the data holder without adversary I, the data can be reconstructed. Such combination is $C^m_{n(1-p_m)}$. The total combinations of retrieved holders are C^m_n. Therefore, the probability of successfully retrieving the data upon the first request is

$$\frac{C^m_{n(1-p_m)}}{C^m_n} = \frac{(n(1-p_m))!(n-m)!}{n!(n(1-p_m)-m)!}.$$

Proposition 2. If majority voting is used for determining the final result, the probability that the voting result is valid in a round is

$$\frac{C^u_{n'} + C^{u-1}_{n'} C^1_{n-n'} + \cdots + C^{\lfloor \frac{u}{2} \rfloor + 1}_{n'} C^{\lceil \frac{u}{2} \rceil - 1}_{n-n'}}{C^u_n},$$

where $n' = n(1-p_m)$.

Proof. The proof is straightforward.

Table 2 Comparisons between data distribution schemes

Scheme	Security	Dependability	Overhead
Individual	yes	no	no
Individual⇒RepS	same	improved	increased
RepS⇒SSBS	improved	improved	decreased
SSBS⇒HybridS	same	same	decreased

Table 3 Quantitative analysis between data distribution schemes[a)]

Scheme	Storage Overhead	Communication Overhead	Computation Overhead
RepS	$n\|DTa_i\|$	$cn\|DTa_i\|$	AE
SSBS	$n\|DTb_i\|$	$cn\|DTb_i\|$	nmME $+n$SE
HybridS	$n(\frac{\|E\|}{m}+\|RSK\|)$	$cn(\frac{\|E\|}{m}+\|RSK\|)$	$2nm$ME$+n$SE

a) ME: Modular exponential; SE: symmetric encryption; AE: asymmetric encryption; $|DTa_i| = |\{\{Data_i\|MAC\}_{RSK} \|\{RSK\}_{PK_u}\}|$; $|DTb_i| = |\{\{Data_i\|MAC\}_{RSK}\|RSK\}| = |E| + |RSK|$; c is the communication cost for 1 bit.

5.3 Dependability analysis

Fault-tolerant:

Proposition 3. The (m, n) HybridS scheme can reconstruct original data if the number of failure nodes in n is smaller than threshold value m. The probability of guaranteeing data reconstruction in the presence of random Byzantine failure is

$$P_{\text{fault-tolerant}} = 1 - \sum_{m \leqslant t \leqslant n} C_n^t p_f^t (1-p_f)^{n-t},$$

where p_f is the probability that a node has random failure and $m \leqslant n$.

Proof. Straightforward.

Figure 1 depicts the analysis results of some (m, n) parameters.

Compromise-resilient:

Proposition 4. Adversaries have to compromise enough nodes in the UWSNs to reveal the original data and damage the confidentiality. For the (m, n) HybridS Scheme, the adversaries have to compromise m nodes from n nodes. If the probability that one node is compromised is p_c, the probability of guaranteeing data confidentiality in the presence of node compromise is

$$P_{\text{compromise-resilient}} = 1 - \sum_{m \leqslant t \leqslant n} C_n^t p_c^t (1-p_c)^{n-t},$$

where p_c is the probability that a node is compromised and $m \leqslant n$.

Proof. Straightforward.

6 Coding method for checking data integrity

In previous presentations, sensor node distributes encoded data to neighbor nodes in a secure and dependable manner. The next problem that arises naturally is how to check the data integrity in a lightweight fashion. As the naive hash based scheme is not desirable and distributed public verification is required for dependability, in this section, we propose a lightweight scheme based on our summarized generic technique.

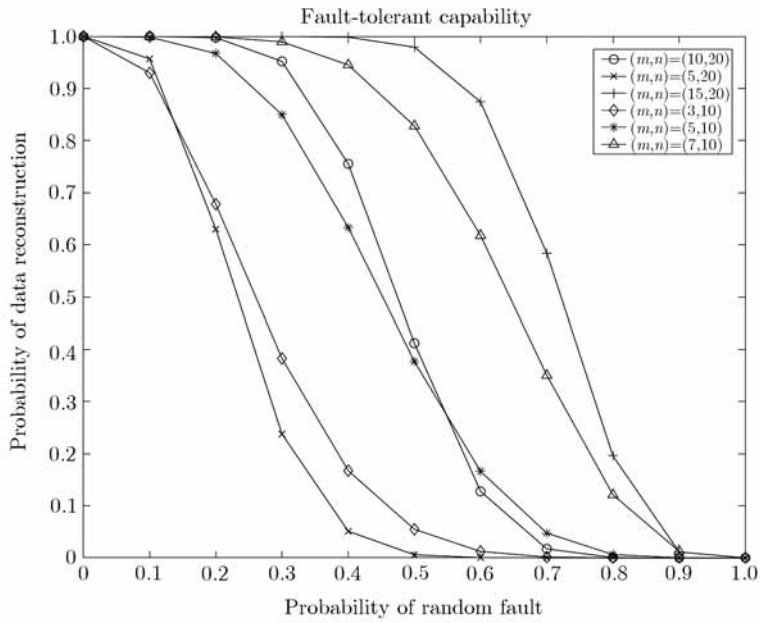

Figure 1 Probability of data reconstruction in presence of random fault.

6.1 TGLC coding method

We employ a two-granularity linear code to check distributed data integrity without original data in a lightweight manner.

Definition 1. Intra-data linear code (Intra-code) is calculated in data using algebraic function:

$$\mathcal{G}_\alpha(x) = \sum_{i=1}^{k} \alpha^{i-1} x_i, \tag{1}$$

where each datum x has k symbols, denoted by x_i. The symbol length is f. α is a primitive number in $GF(2^q)$. Besides, $q = f$ and $k \leqslant 2^q - 1$.

Definition 2. Inter-data linear code (Inter-code) is calculated among data using algebraic function:

$$\mathcal{P}(y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} \beta^{i-1} y_i, (i \in [1, n]), \tag{2}$$

where each datum $y_i (i \in [1, n])$ has k symbols. The symbol length is f, and β is a primitive number in $GF(2^q)$. Also, $q = f$ and $n \leqslant 2^q - 1$.

Theorem 1. The Intra-code's Inter-code equals Inter-code's Intra-code.

Proof. Consider data $S_1, S_2, \ldots, S_n$, each of which has k symbols, denoted by $S_j = \{x_{1,j}, x_{2,j}, \ldots, x_{k,j}\}$. The symbol length is f. α and β are primitive numbers in $GF(2^q)$. Besides, $q = f$ and $k, n \leqslant 2^q - 1$.

(1) The Inter-code of $S_1, S_2, \ldots, S_n$ is $P = \sum_{j=1}^{n} \beta^{j-1} S_j$. Denote the k symbols of P by $P_i, i \in [1, k]$. Then

$$P_i = \sum_{j=1}^{n} \beta^{j-1} x_{i,j}, i \in [1, k].$$

(2) The Inter-code's Intra-code is

$$\mathcal{G}_\alpha(P) = \sum_{i=1}^{k} \alpha^{i-1} P_i = \sum_{i=1}^{k} \alpha^{i-1} \left(\sum_{j=1}^{n} \beta^{j-1} x_{i,j} \right).$$

(3) The Intra-code of data $S_j, j \in [1,n]$ is $\mathcal{G}_\alpha(S_j), j \in [1,n]$. We have

$$\mathcal{G}_\alpha(S_j) = \sum_{i=1}^{k} \alpha^{i-1} x_{i,j}, (j \in [1,n]).$$

(4) The Inter-code of Intra-code $\mathcal{G}_\alpha(S_j), j \in [1,n]$ is

$$\mathcal{P}(\mathcal{G}_\alpha(S_1), \ldots, \mathcal{G}_\alpha(S_j)) = \sum_{j=1}^{n} \beta^{j-1} \mathcal{G}_\alpha(S_j) = \sum_{j=1}^{n} \beta^{j-1} \Big(\sum_{i=1}^{k} \alpha^{i-1} x_{i,j} \Big).$$

(5) Due to the property of field and linear coding in $GF(2^q)$, we have

$$\sum_{j=1}^{n} \beta^{j-1} \Big(\sum_{i=1}^{k} \alpha^{i-1} x_{i,j} \Big) = \sum_{i=1}^{k} \alpha^{i-1} \Big(\sum_{j=1}^{n} \beta^{j-1} x_{i,j} \Big).$$

Intra-code is calculated by data holders; Inter-code is calculated by data distributors and distributed to verifiers. For simplicity of description, we call Intra-code diGest and Inter-code Parity in the rest of the paper.

By choosing multiple α, the single Intra-code can be extended to multiple digests. Similarly, by choosing multiple β, the single Inter-code can be extended to multiple parities.

Two Granularity Linear Code used for checking data integrity has two advantages as follows: 1) The checking is in a distributed manner. The verifier who holds the parity can launch checking by requesting the digests from the data holders, and then checking the integrity equation (in Theorem 1). The distributed checking thus is possible, and is more dependable. 2) Lightweight checking. The involved communication is only digests that are much shorter than the original data or hash values. The resulting computation is only algebraic functions that are computed much faster than hashing functions, at both verifiers and holders.

6.2 TGLC based integrity checking

The steps for integrity checking are listed as follows:

(1) A distributor generates u parities using u randomly selected numbers: $\beta_1, \beta_2, \ldots, \beta_u$ using eq. (2).

(2) The distributor distributes u parities to u verifiers.

(3) The verifiers who received the parity $\{P_j \| \beta_j\}$ can launch data integrity checking. That is, v randomly selected numbers: $\alpha_1, \alpha_2, \ldots, \alpha_v$ as challenges and send them to distributed data holders.

(4) Each distributed data holder computes v digests using $\alpha_1, \alpha_2, \ldots, \alpha_v$ by eq. (1) and sends them back to the verifier.

(5) The verifier computes the parities of the returned digests, calculates the digests of the parity, and checks whether the equations hold:

$$\mathcal{G}_{\alpha_i}(P_j) \stackrel{?}{=} \sum_{t=1}^{n} \beta_j^{t-1} \mathcal{G}_{\alpha_i}(S_t), i \in [1,v].$$

If the v checking equations hold, the data integrity is maintained.

Figure 2 depicts the generation of the parities.

Implementation consideration: The Reed-Solomon codes, Secret Sharing and TGLC can all be efficiently implemented in the same Galois field library. Thus, the implementation code can share the same library. For example, Plank and Greenan et al. [23, 24] have already released fast Galois field library for arithmetic in $GF(2^8)$, $GF(2^{16})$ and $GF(2^{32})$.

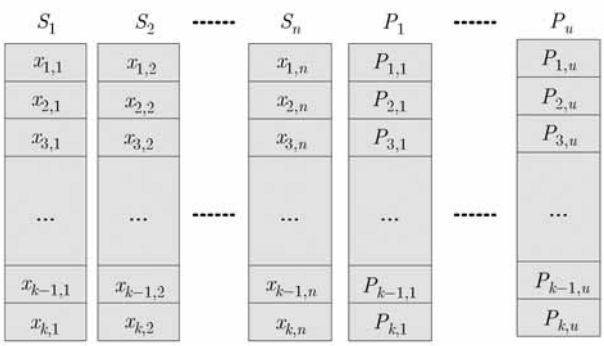

Figure 2 u parities are generated from n data.

6.3 Analysis of integrity checking

Lemma 1. If $v = 1$ in the proposed scheme, only one digest is returned from each data holder. Suppose one holder (denoted by d) holds data S. Suppose S is modified into S'. Once the holder d receives the challenge $\{\alpha \| v\}$ ($\alpha \in GF(2^q)$, $v = 1$), it returns digest $\mathcal{G}_\alpha(S')$. Suppose the digest of S for α is $\mathcal{G}_\alpha(S)$. The probability that the datum S is modified but the digest of data remains unchanged is

$$\Pr\{\mathcal{G}_\alpha(S) = \mathcal{G}_\alpha(S'), S \neq S'\},$$

denoted by $\Pr_1$. That is,

$$\Pr_1 = \frac{(2^f + 1)^k - k * 2^f - 1}{(2^{fk} - 1)2^f}.$$

Proof. Suppose the data S has k symbols, denoted by $x_0, x_1, \ldots, x_k$. The symbol length is f. (In fact, we have $q = f$ and $k \leqslant 2^q - 1$.) If $v = 1$, only one digest is returned, which is generated by

$$\mathcal{G}_\alpha(S) = x_0 + \alpha x_1 + \cdots + \alpha^{k-1} x_k.$$

(1) If only one symbol in data is changed, the digest of data will be changed. For example, suppose $x_i (i \in [1, k]))$ is changed to x'_i. Then $\mathcal{G}_\alpha(S') \neq \mathcal{G}_\alpha(S)$, where $\mathcal{G}_\alpha(S') = x_0 + \cdots + \alpha^{i-1} x'_i + \cdots + \alpha^{k-1} x_k$.

(2) If two symbols are changed, the digest may be unchanged. For example, suppose x_{i_1} and x_{i_2} ($i_1, i_2 \in [1, k]$) are changed to x'_{i_1} and x'_{i_2}. The total number of pairs is $2^f * 2^f$, in which 2^f pairs have the digest $\mathcal{G}_\alpha(S)$. Because for each x'_{i_1}, there always exists an x'_{i_2} such that

$$\alpha^{i_1 - 1} x'_{i_1} + \alpha^{i_2 - 1} x'_{i_2} = \alpha^{i_1 - 1} x_{i_1} + \alpha^{i_2 - 1} x_{i_2}.$$

That is, the digest keeps unchanged. The number of such pairs is 2^f, so the total occasions in which two symbols are modified but the digest is unchanged is $C_k^2 2^f$.

(3) Similarly to (2), if t ($t \in [3, k]$) symbols are changed, the occasions in which t symbols are modified but the digest is still equal to $\mathcal{G}_\alpha(S)$ is $C_k^t 2^{(t-1)f}$.

(4) The total number of occasions where the data are modified is $(2^f)^k - 1$. (Each symbol has 2^f modification occasions, and the data have k symbols in total. Besides, the occasion without modification is excluded.) Therefore, the probability that data are modified but the digest is unchanged is

$$\Pr_1 = \frac{C_k^2 2^f + C_k^3 2^{2f} + \cdots + C_k^k 2^{(k-1)f}}{2^{fk} - 1} = \frac{(2^f + 1)^k - k2^f - 1}{(2^{fk} - 1)(2^f)}.$$

Figure 3 depicts the generation of the parities.

In fact, $\Pr_1$ can be estimated by 2^{-f}, because

$$\begin{aligned}\Pr_1 &= \frac{(2^f + 1)^k - k2^f - 1}{(2^{fk} - 1)(2^f)} = \frac{(1 + 2^{-f})^k - k2^{-k} - 2^{-kf}}{1 - 2^{-fk}}(2^{-f}) \\ &\approx ((1 + 2^{-f})^k - k2^{-k})(2^{-f}) = (1 + 2^{-f})^k 2^{-f} - k2^{-kf} \approx (1 + 2^{-f})^k 2^{-f} \approx 2^{-f}.\end{aligned}$$

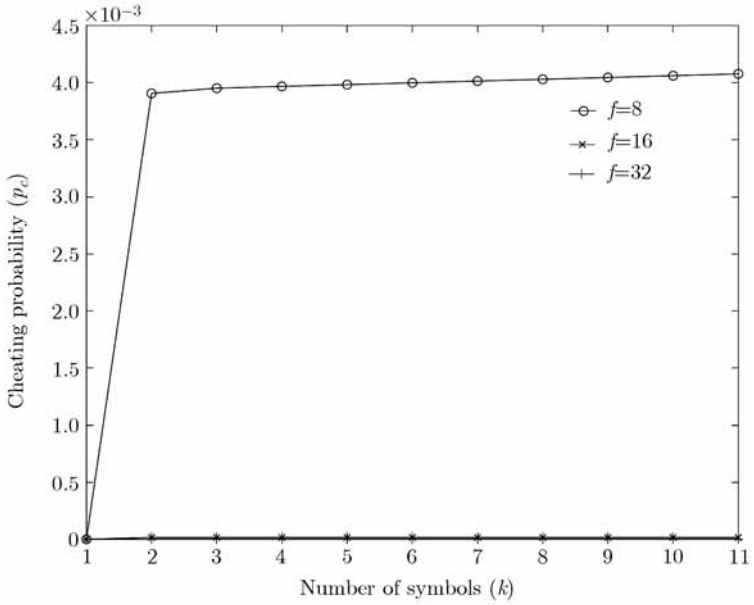

Figure 3 The probability that the data are modified but the digest is unchanged corresponds with the number of symbols and the length of symbol.

Lemma 2. If $v \geqslant 2$ in proposed scheme, v digests are expected to return from each data holder. Suppose one holder (denoted by d) holds data S. Suppose S is modified into S'. Once the holder d receives the challenge $\alpha_1, \ldots, \alpha_v$, it returns digests $\mathcal{G}_{\alpha_i}(S'), (i = 1, \ldots, v)$. The digests of S for $\alpha_i, (i = 1, \ldots, v)$ are $\mathcal{G}_{\alpha_i}(S), (i = 1, \ldots, v)$. The probability that the data S are modified but the digests of data remain unchanged is

$$\Pr\{\mathcal{G}_{\alpha_i}(S) = \mathcal{G}_{\alpha_i}(S'), (i = 1, \ldots, v), S \neq S'\},$$

which is denoted by $\Pr_2$. That is,

$$\Pr_2 = \frac{C_k^{v+1}2^f + C_k^{v+2}2^{2f} + \cdots + C_k^k 2^{(k-v)f}}{2^{fk} - 1}.$$

Proof. According to Lemma 1, only when the number of modified symbols in data is larger than v, does the occasion that data is modified but v digests are unchanged become possible.

(1) If $v + 1$ symbols are modified, without loss of generality, we denote them by $x_1, x_2, \ldots, x_{v+1}$. The difference between original data and modified data is $\Delta x_1, \Delta x_2, \ldots, \Delta x_{v+1}$. We have

$$\begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{k-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{k-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha_u & \alpha_v^2 & \cdots & \alpha_v^{k-1} \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{3}$$

The coefficient matrix is a Vandermonde matrix with linear independence, and dimension v. Thus, 2^f solutions exist for eq. (3). So, the probability that the $v + 1$ symbols are modified but the v digests remain unaltered is $C_k^{v+1}2^f$.

(2) Similarly to 1), the total occasions where the data are modified but the v digests remain unchanged is

$$C_k^{v+1}2^f + C_k^{v+2}2^{2f} + \cdots + C_k^k 2^{(k-v)f}.$$

Because the total modification occasions are $(2^f)^k - 1$ (same reason as that for Lemma 1(4)), the resultant probability that the data are modified but the v digests remain unaltered is

$$\Pr_2 = \frac{C_k^{v+1}2^f + C_k^{v+2}2^{2f} + \cdots + C_k^k 2^{(k-v)f}}{2^{fk} - 1}.$$

When $k >> v$, the $\Pr_2$ can be estimated by 2^{-vf}, because

$$\frac{C_k^{v+1}2^f + C_k^{v+2}2^{2f} + \cdots + C_k^k 2^{(k-v)f}}{2^{fk}-1} = \frac{C_k^{v+1}2^{-(k-1)f} + C_k^{v+2}2^{-(k-2)f} + \cdots + C_k^k 2^{-vf}}{1-2^{-fk}}$$
$$\approx C_k^{v+1}2^{-(k-1)f} + C_k^{v+2}2^{-(k-2)f} + \cdots + C_k^k 2^{-vf} \approx 2^{-vf}.$$

Lemma 3. Suppose $u = 1$. Then the probability that the data S_i's digest is changed but the parity remains unchanged is

$$\Pr_3 = \frac{(2^f+1)^n - n*2^f - 1}{(2^{fn}-1)2^f}.$$

Proof. The proof is similar to that of Lemma 1, and similarly, $\Pr_3 \approx 2^{-f}$.

Lemma 4. Suppose u parities are distributed in the proposed scheme. The probability that the returned digests are changed but the u parities still remain unchanged is

$$\Pr_4 = \frac{C_n^{u+1}2^f + C_n^{u+2}2^{2f} + \cdots + C_n^n 2^{(n-u)f}}{2^{fn}-1}.$$

Proof. The proof is similar to that of Lemma 2, and similarly, when $n >> u$, $\Pr_4 \approx 2^{-uf}$.

Definition 3. The ealse negative rate (FNR) ($\mathbb{R}$) is the probability that distributed data are modified, but the checking result shows no data modification due to the satisfaction of checking equations.

If the checking equations are not satisfied, we can draw the conclusion that the data is modified. If the checking equations are satisfied, we claim that the data are not modified, but this claim is correct only for the probability of $1-\mathbb{R}$. The FNR is possible due to the parameter selections. To ensure that the $\mathbb{R}$ is small enough, we analyze $\mathbb{R}$ in different parameter settings.

Theorem 2. Suppose one parity ($u = 1$) and one digest ($v = 1$) are employed. Then the proposed scheme has $\mathbb{R} = C_{n-1}^1\Pr_1 + C_{n-1}^2(\Pr_1^2 + (1-\Pr_1)^2\Pr_3) + \sum_{z=3}^{n-1} C_{n-1}^z(\Pr_1^z + \sum_{i=2}^z C_z^i \Pr_1^{z-i}(1-\Pr_1)^i\Pr_3)$.

Proof. $v = 1$ indicates that only one digest is returned from the data holders; $u = 1$ implies that the verifier holds one parity. The verifier receives $n-1$ digests from $n-1$ responding nodes, as the verifier itself holds one datum and can generate the digest. Upon receiving such $n-1$ digests, the verifier will check the integrity equation. Recall that the checking equation is $\mathcal{G}_\alpha(P_1) \stackrel{?}{=} \mathcal{P}_1(\mathcal{G}_\alpha(S_1), \ldots, \mathcal{G}_\alpha(S_n))$ in this case ($u = v = 1$).

If the modified data pass the checking equation, they must belong to one of two types: (Type-I) The digests of the modified data equal the digests of the original data; (Type-II) Some digests of the modified data are different but the parity of all digests is the same as expected. Next, we categorize different modification occasions, and analyze such two types for each occasion.

(1) Suppose one datum in $n-1$ is modified but undetected. If Type-I occurs (the digest is unchanged), the probability is $\Pr_1$ according to Lemma 1. Type-II cannot occur because one changed digest definitely results in the change of parity. Therefore, the probability that one datum is modified but the checking equation is satisfied is $C_{n-1}^1\Pr_1$.

(2) Suppose two data in $n-1$ are modified. If Type-I occurs, the probability that two digests are unchanged is $\Pr_1^2$. If Type-II occurs, the probability that two digests are changed but their parity is unchanged is $(1-\Pr_1)^2\Pr_3$. Therefore, the probability that two data are modified but the checking equation is satisfied is $C_{n-1}^2(\Pr_1^2 + (1-\Pr_1)^2\Pr_3)$.

(3) Suppose z ($z \in [3, n-1]$) data in n are modified. Similarly, if Type-I occurs (all z digests are unchanged), the probability that z is unchanged is $\Pr_1^z$. When Type-II occurs, the probability will be further analyzed as follows:

Suppose i in z digests are changed but the parity is the same. Then its probability is $C_z^i\Pr_1^{z-i}(1-\Pr_1)^i\Pr_3$. Also, at least two digests must be changed to make the parity unchanged, so $i \in [2, z]$. The probability that z digests are changed but parity is same can be computed by summation. That is, $\sum_{i=2}^z C_z^i\Pr_1^{z-i}(1-\Pr_1)^i\Pr_3$.

Therefore, the probability that z data are modified but undetected is

$$C_{n-1}^{z}\left(\mathrm{Pr}_1^z+\sum_{i=2}^{z}C_z^i\mathrm{Pr}_1^{z-i}(1-\mathrm{Pr}_1)^i\mathrm{Pr}_3\right), z\in[3,n-1].$$

(4) In summary, the probability that the data is modified but undetected for proposed scheme with $u=v=1$ is the summation of (1), (2) and (3). That is,

$$\mathbb{R}=C_{n-1}^1\mathrm{Pr}_1+C_{n-1}^2(\mathrm{Pr}_1^2+(1-\mathrm{Pr}_1)^2\mathrm{Pr}_3)+\sum_{z=3}^{n-1}C_{n-1}^{z}\left(\mathrm{Pr}_1^z+\sum_{i=2}^{z}C_z^i\mathrm{Pr}_1^{z-i}(1-\mathrm{Pr}_1)^i\mathrm{Pr}_3\right).$$

Theorem 3. Suppose u parities and one digest ($v=1$) are used. Then the proposed scheme has $\mathbb{R}=C_{n-1}^1\mathrm{Pr}_1+\sum_{i=2}^{u}C_{n-1}^i\mathrm{Pr}_1^i+\sum_{z=u+1}^{n-1}C_{n-1}^z(\mathrm{Pr}_1^z+\sum_{i=u+1}^{z}C_z^i\mathrm{Pr}_1^{z-i}(1-\mathrm{Pr}_1)^i\mathrm{Pr}_4)$.

Proof. We use the same setting in that of Theorem 2, and the proof is similar to Theorem 2.

(1) The probability that one datum in n is modified but undetected is $C_{n-1}^1\mathrm{Pr}_1$.

(2) Suppose z ($z\in[2,u]$) data are modified but undetected. If Type-I occurs, the probability that z digests are unchanged is Pr_1^z. When $z<u$, the parities of any changed z digests will be different from the original one. Hence Type-II cannot occur. The probability that z data in n are modified without detection is $C_{n-1}^z\mathrm{Pr}_1^z$.

(3) Suppose z ($z\in[u+1,n-1]$) data are modified but undetected. If Type-I occurs, the probability that z digests are unchanged is Pr_1^z. If Type-II occurs, the occasions are further analyzed in the following, which is similar to (3) in Theorem 2.

Suppose i in z digests are changed but the u parities are the same. The probability that i digests are changed but the u parities are the same is $C_z^i\mathrm{Pr}_1^{z-i}(1-\mathrm{Pr}_1)^i\mathrm{Pr}_4$, and $i\in[u+1,z]$. We thus can compute the probability that z digests are changed but the u parities are the same by summation, which is $\sum_{i=u+1}^{z}C_z^i\mathrm{Pr}_1^{z-i}(1-\mathrm{Pr}_1)^i\mathrm{Pr}_4$.

Therefore, the probability that z data are modified but undetected is: $C_{n-1}^z(\mathrm{Pr}_1^z+\sum_{i=u+1}^{z}C_z^i\mathrm{Pr}_1^{z-i}(1-\mathrm{Pr}_1)^i\mathrm{Pr}_4)$.

(4) In summary, the probability that the data are modified but undetected for proposed scheme with parameters u and $v=1$ is

$$\mathbb{R}=C_{n-1}^1\mathrm{Pr}_1+\sum_{i=2}^{u}C_{n-1}^i\mathrm{Pr}_1^i+\sum_{z=u+1}^{n-1}C_{n-1}^{z}\left(\mathrm{Pr}_1^z+\sum_{i=u+1}^{z}C_z^i\mathrm{Pr}_1^{z-i}(1-\mathrm{Pr}_1)^i\mathrm{Pr}_4\right).$$

Theorem 4. Suppose u parities and v digests are used. Then the proposed scheme has $\mathbb{R}=C_{n-1}^1\mathrm{Pr}_2+\sum_{i=2}^{u}C_{n-1}^i\mathrm{Pr}_2^i+\sum_{z=u+1}^{n-1}C_{n-1}^z(\mathrm{Pr}_2^z+\sum_{i=u+1}^{z}C_z^i\mathrm{Pr}_2^{z-i}(1-\mathrm{Pr}_2)^i\mathrm{Pr}_4)$.

Proof. The proof is similar to that of Theorems 2 and 3.

Theorem 5. The FNR $\mathbb{R}$ decreases exponentially corresponding to appropriately selected parameters.

Proof. As in Lemmas 1–4, $\mathrm{Pr}_1=\mathrm{Pr}_3\approx 2^{-f}$, $\mathrm{Pr}_2\approx 2^{-vf}$, $\mathrm{Pr}_4\approx 2^{-uf}$. According to the FNR $\mathbb{R}$ in Theorems 2–4, $\mathbb{R}$ decreases exponentially from the selection of $u=v=1$ to $u>1, v=1$ because $\mathrm{Pr}_3>\mathrm{Pr}_4$ exponentially, and from the selection of $u>1, v=1$ to $u,v>1$ because $\mathrm{Pr}_1>\mathrm{Pr}_2$ exponentially.

In previous analysis, we only discussed one-round checking situation. If the checking rounds increase in number, the FNR will decrease exponentially. As is proven in Theorem 6, FNR can be further decreased by adding checking rounds.

Theorem 6. The FNR after r rounds of integrity checking is $\mathbb{R}^r$, where $\mathbb{R}$ is FNR for one round.

Proof. Suppose the proposed scheme checks r rounds. If in one of them the checking equations are not satisfied, the proposed scheme can draw the conclusion that the data are modified. Such a conclusion has zero false positive rate. If the checking equations are satisfied in all r rounds, the proposed scheme claims that the data are not modified. The FNR after r rounds is $\mathbb{R}^r$. The claim that the data integrity is maintained is correct with the confidence $1-\mathbb{R}^r$, which converges to 1 with the addition of r.

7 Conclusions

In this paper, we have proposed a secure and dependable data distributed storage scheme for in-networking stored sensing data in UWSNs. To achieve the maximum of accessible data upon retrieval, we take advantages of secret sharing and Reed-Solomon code, without loss of computational security and with low communication and storage overhead. The extensive analysis on performance, security and dependability verified that our scheme performs well even in the presence of node compromise and Byzantine failure. Besides it remains efficient in terms of storage, communication and computation owing to shortened distributed data size.

Furthermore, we presented a generic coding method called Two Granularity Linear Code to efficiently check data integrity. It consists of Intra-code and Inter-code that can be computed commutatively. Such a property provides inner linkage of distributed data via digests and parities with small size. It can be applied to verify data integrity publicly in a distributed manner without original data. By making use of TGLC, we have thus proposed a lightweight scheme for verifying distributed data integrity. The extensive analysis has verified that the false positive of checking result is zero and the false negative vanishes exponentially with appropriate coding parameters and checking rounds.

Acknowledgements

This work was supported by the Special Fund for Basic Scientific Research of Central Colleges, China University of Geosciences (Wuhan) (Grant No. 090109), the Science and Research Start-Up Project for the Recruit Talent of China University of Geosciences (Wuhan) (Grant No. 20090113). Part of the work was done in Department of Electrical and Computer Engineering, Illinois Institute of Technology, USA.

References

1 Thuraisingham B. Secure sensor information management and mining. Signal Process Mag IEEE, 2004, 21: 14–19
2 Pietro R D, Mancini L V, Soriente C, et al. Catch me (if you can): Data survival in unattended sensor networks. In: Proc. of 6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'08), Hong Kong, China, 2008. 185–194
3 Diao Y, Ganesan D, Mathur G, et al. Rethinking data management for storage-centric sensor networks. In: Proc. of the Third Biennial Conference on Innovative Data Systems Research (CIDR'07), Asilomar, CA, 2007. 22–31
4 Girao J, Westhoff D, Mykletun E, et al. Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. Ad Hoc Networks, Elesevier, 2007, 5: 1073–1089
5 Zeinalipour-Yazti D, Kalogeraki V, Gunopulos D, et al. Towards in-situ data storage in sensor databases. In: Proc. of 10th Panhellenic Conference on Informatics (PTI'05), LNCS 3746, Volos Greece, 2005. 36–46
6 Osrovsky R, Yung M. How to withsand mobile virus attack. In: Proc. of PODC, Montreal, Quebec, Canada, 1991. 51–59
7 Ma D, Soriente C, Tsudik G. New adversary and new threats: Security in unattended sensor networks. IEEE Network, 2009, 23: 43–48
8 Ma D, Tsudik G. Forward-secure sequential aggregate authentication. In: IEEE Symposium on Security and Privacy (IEEE S&P'07), Oakland, May 2007. 86–91
9 Ganesan D, Greenstein B, Estrin D, et al. Multiresolution storage and search in sensor networks. ACM Trans Stor, 2005, 1: 277–315
10 Mathur G, Desnoyers P, Ganesan D, et al. Ultra-low power data storage for sensor networks. In: Proc. of the Fifth International Conference on Information Processing in Sensor Networks (IPSN'06), Nashville, Tennessee, USA, 2006. 374–381
11 Banerjee A, Mitra A, Naijar W, et al. Rise-co-s: High performance sensor storage and co-processing architecture. In: Proc. of Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Natowkrs, (SECON'05), Santa Clara, California, 2005. 1–12
12 Blundo C, Santix A D, Herzberg A, et al. Perfectly-secure key distribution for dynamic conferences. In: Proc. of CRYPTO'92, Santa Barbara, California, USA, 1992. 471–486
13 Bhatnagar N, Miller E L. Designing a secure reliable file system for sensor networks. In: Proc. of the 2007 ACM Workshop on Storage Security and Survivability (Storage'07), Alexandria, Virginia, 2007

14 Girod L, Lukac M, Trifa V, et al. The design and implementation of a self-calibrating distributed acoustic sensing platform. In: Proc. of ACM SenSys'06, Baulder Colorado, USA, 2006
15 Inel corporation. Intel mote 2. http://www.intel.com/research/.
16 Shamir A. How to share a secret. Comm ACM, 1979, 22: 612–613
17 Reed S, Solomon G. Polynomial codes over certain finite. J Soc Indust Appl Math, 1960, 8: 300–304
18 Lou W, Liu W, Fang Y. Spread: Enhancing data confidentiality in mobile and hoc networks. In: Proc. of IEEE INFOCOM'04, Hong Kong, China, 2004. 2404–2413
19 Cachin C, Tessaro S. Optimal resilience for erasure-coded byzantine distributed storage. In: Proc. of Dependable Systems and Networks (DSN'06), Washington, DC, USA, 2006. 115–124
20 Aguilera M K, Janakiraman R, Xu L. Using erasure codes efficiently for storage in a distributed systems. In: Proc. of DSN'05, Palo Alto, CA, USA, 2005. 336–345
21 Belenkiy M. Disjunctive multi-level secret sharing. Cryptology ePrint Archive, Report 2008/018, 2008, http://eprint.iacr.org/
22 Benaloh J. Secret sharing homomorphisms: keeping shares of a secret secret. In: Proc. of Crypto'86, Santa Barbara, California, USA, 1987. 251–260
23 Plank J S. Fast galois field arithmetic library in c/c++. Tech. Report UT-CS-07-593, 2007
24 Greenan K M, Miller E L, Schwarz T. Analysis and construction of galois fields for efficient storage reliability. Tech. Report Number SSRC-07-09, UCSC, Storage Systems Research Center, August 2007