Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 2785960, 22 pages
https://doi.org/10.1155/2020/2785960

Research Article

Rank-Based Report Filtering Scheme (RRFS) for Verifying Phoney Reports in Wireless Sensor Networks

Gayathri Santhosh and Yogesh Palanichamy

Department of Information Science and Technology, College of Engineering, Anna University, Guindy, Chennai 600025, India

Correspondence should be addressed to Gayathri Santhosh; m.gayath@yahoo.com

Received 11 February 2020; Revised 17 August 2020; Accepted 20 August 2020; Published 15 September 2020

Academic Editor: Pierre-Martin Tardif

Wireless sensor networks (WSNs) are open to false data injection attack when they are deployed in hostile scenarios. Attackers can easily deceive the sink by compromising sensing nodes or by injecting phoney data into the network. Such attacks can deplete the energy resources of the network by providing wrong information which in turn can affect the proper network functioning or sometimes can shut the network from further functioning. The existing schemes that deal with this problem focus on only a few aspects of the false data injection attack. To resolve this problem, we propose a Rank-based Report Filtering Scheme (RRFS), a holistic and group verification scheme for the identification of compromised nodes and the filtering of false data injected into the network. The proposed scheme verifies report among clusters, en-routers, and sink. Hence, the RRFS, a holistic scheme that is composed of three-tier verifications, successfully rejects the false data before the attackers falsify the whole environment, and this makes the system unique. Reliability Index (RI) is calculated by the nodes for fellow cluster members, and the cluster head (CH) provides the score for a node based on its RI. This, in turn, strengthens the scheme by assisting the en-routers to detect the compromised nodes. The RRFS scheme has been verified and validated by extensive simulation and meticulous performance evaluation of filtering efficiency and energy consumption against various schemes. The scheme gives high filtering efficiency against the multiple compromised nodes and also improves the network's lifespan. The sustainability of RRFS against numerous attacks that are launched in the sensor environment is thoroughly investigated.

1. Introduction

Wireless sensor network is a collection of sensor nodes that works together to sense various physical parameters by monitoring the given domain [1]. Each node has various sensors, actuators, a wireless transceiver, a microcontroller, and power sources. Recent developments that have taken place in the areas of wireless communication and microelectromechanical system (MEMS) have led to the deployment of distributed wireless sensing systems. WSNs are widely used in unfavourable terrains like forests, deserts, and battlefields [2]. In recent times, they gain importance in urban areas. As of now, the applications of WSNs include healthcare, military, environmental, smart cities [3], and other regions. WSNs are inherently vulnerable since they are wireless and deployed in remote locations where physical monitoring is difficult.

The major attacks launched against WSNs are Sybil attack [4], wormhole attack [5], DoS attack [6], black hole attack [7], eavesdropping [8], jamming [9] etc. Out of all the possible attacks, false data injection attack is the most critical attack. Adversary compromises the sensor node [10] and pushes false or malicious data into the network in such a way that the operators of the network would not be able to realise that the false/malicious data have been injected inside the network. Event-based report generation is compromised with the false data injection, which misleads the sink to initiate a wrong activity. It also depletes the energy of the nodes by making them forward unnecessary reports to the sink.

Several verification schemes have been proposed for filtering the injected false data in the network [11–18]. We have observed that many of these schemes are prone to node compromise attack [19] and selective forwarding attack [20].

False report sometimes escaped the verification and increases communication overhead. An effective verification scheme is a need to verify false data by organizing nodes into cluster on their proximity to the events, and one among the cluster nodes must collect the report and forwards the report to sink. En-route filtering must be done to detect a compromised node in the network, which in turn also increases the lifespan of the network by avoiding unnecessary verification by the next hop node. Both the sink and the en-routers shall carry out the verification of the report to ensure authenticity.

Rank-based Report Filtering Scheme (RRFS) for verifying phoney reports in wireless sensor networks is proposed in this paper to filter the false data injected into the network. The methodology followed in RRFS are summarised as follows:

(i) RRFS organizes the nodes of the network into multiple clusters, and the reports are forwarded through different paths that are provided by different clusters. Clusters are dynamic, i.e., cluster members and CH are different for every session, and the session corresponds to an event sensed by the nodes. RRFS ensures the reliable End-to-End (E2E) delivery of reports by forwarding the reports in multiple paths in a secure way to the sink

(ii) Detecting nodes generate three different reports for the same event. Private Reports (PR) are generated using private keys for sink verification. Cluster members generate Intracluster Reports (IR) using a selfish key (S_k) which is verified by cluster members. IR plays a vital role in the calculation of the Reliability Index (RI) of the nodes. Nodes generate Intercluster Reports (ICR) using a guest key (G_k) for en-router verification. Hence, the en-routers can perform cross-verification and ensure that phoney reports will be filtered with a very high probability even if some portion of the network fails in preventing them

(iii) Another parameter that plays a major role in the proposed scheme is the score (S) which is calculated by the CH based on RI of the constituent nodes. RI is computed using three parameters, namely honesty (H), kindness (K), and strength (St) that are assigned to the nodes based on their characteristics. After ICR verification fails, ECH asks for score from downstream CH. If S is low, the forwarding cluster drops the false data and reports to the sink about the node compromise attack. This aids the other nodes from unnecessarily forwarding phoney reports and thus increasing the longevity of the network.

The proposed work is organized as follows. Section 2 discusses the related works. Section 3 explains the system model and assumptions. Section 4 describes the preliminaries used in the proposed scheme. Section 5 explains the methodology followed in the proposed scheme which includes five phases, namely predeployment phase, postdeployment phase, event detection phase, Reliability Index (RI), Cluster Score (CS), and report verification phase. Section 6 discusses the performance evaluation by comparing RRFS with other schemes. Section 7 outlines how RRFS withstands the attacks that usually happen in an unattended environment. Section 8 concludes the proposed work and future ideas.

2. Related Works

Several verification schemes have been formulated for filtering the injected false data in WSN. Statistical En-Route Filtering (SEF) of injected false data [21] is one of the earliest filtering schemes. SEF has a nonoverlapping global key pool partition in which the authentication keys are stored. Reports are generated based on events using these keys. Message Authentication Code (MAC) has been generated using the keys for an event. Sink verifies MAC using the key pool. En-router checks the report if it shares the key partition and drops the report if there is a mismatch in the MAC. This scheme will not work if the attacker compromises the key pool. In Interleaved Hop-by-hop Authentication (IHA) scheme, the BS initiates the association process to make the nodes get connected and to establish the pairwise keys [22]. Two reports are generated for each event using pairwise keys with association nodes and the keys that are shared with the sink. A fixed path is needed for report forwarding, and hence, path maintenance and recovery become a tedious process. Secure Ticket-base En-route Filtering (STEF) scheme [23] is based on query-based one-way function. It is a ticket-based concept in which the sink randomly selects the area of interest and sends a query which has a ticket in it. The ticket receiving node forwards the message about an event to the sink through the cluster head (CH). Ticket "C" is exclusively for the CH because "C" is encrypted using the personal key that is stored only between the node and the sink. At a particular time instant, the cluster members generate their reports with their own keys and forward them to the CH. En-router checks the report by calculating the hash value for C and compares that with the "C" which was stored earlier. En-router only looks for the validity of "C" and does not verify the content.

Multipath Interleaved Hop-by-hop Authentication (MIHA) scheme works for disjoint and braided paths [14]. This scheme switches the path if n number of compromised nodes are found in their current route. Multiple paths have been established between the source and the BS. Key distribution takes place for all the paths that are established in the network. The new authentication key is shared between the downstream node and the router. The en-router uses the hash function to calculate the authentication key and to check the authenticity of the report. Route switching is performed by a sink to detect if there are n numbers of attackers in a single path. Periodic messages that are sent by the sink to make the path alive consume extra resources. Attackers make use of multipath technique and exploit the report forwarding nodes, which are located near the base station. Bandwidth-Efficient Cooperative AutheNtication (BEACAN) scheme is a cooperative bit-compressed authentication method for providing bandwidth-efficient authentication [24]. This scheme uses a

collaborative report generation for an event with the help of neighbours. Predefined path is calculated for every session by source nodes. Source node asks cluster member to generate a report for the nodes in the defined path, which makes the scheme complicated. However, this scheme will not work if the whole cluster is compromised.

Dynamic En-route Filtering (DEF) scheme [25] assures that clusters are formed around the events and functions only if the groups are formed. Multiple reports are generated for each round. Keys are disseminated after forwarding the reports to the upstream forwarding nodes for every round. Upstream nodes make use of disclosed keys by validating the received reports. The excess overhead here is due to the increased number of keys and control messages. New control messages triple the delay of the report. Wang et al. have proposed a scheme that focuses on the geographical location of the nodes to endorse whether the report is logical or not [17]. Each node acquires its position and keys and exchanges the information to the nearby node(s). It is easy for the attackers to detect the neighbouring nodes since they store the key index of the neighbouring node. This scheme focuses on finding whether the report generation is legitimate, leaving the verification to the sink. However, it does not analyze the details of the report. In Location-aware End-to-end Data Security [26] (LEDS), the area is divided into multiple cells to bind the location information using a symmetric key. It prevents selective forwarding attack with the unique private key that is shared between the sink and the node. This unique key helps the sink to check the node authenticity. However, the LEDS does not prevent a compromised node from forwarding the report, and hence, it is not suitable for dynamic networks. PKAEF [27] is a location-based scheme, which detects the compromised nodes and prevents them from injecting further false data into the network. This scheme has T threshold limitations, and hence, localization is a tedious process. Seluk et al. proposed Time-based Dynamic Keying and En-route Filtering (TICK) [28] which does not need a key exchange but key changes as a function of the clock. Three modules have been configured in the nodes; namely, Time-based Key Management (TKM), Cryptosystem (CRYPTO), and Filtering-Forwarding (FFWD). TKM is responsible for dynamic key generation. CRYPTO takes care of encryption by dynamic key and forwards the report. A receiving node finds the dynamic key using propagation time, packet transmission time, and packet processing time. FFWD verifies the packet. All these imply that TICK is not suitable for the uncontrolled environment.

Grouping-enhanced Resilient Probabilistic En-route Filtering (GRPEF) [15] is a location-based scheme that uses a multiaxis division approach to derive the keys. Grouping algorithm is activated for covering all the nodes in the network, and reports are generated based on group and axis. An en-router receives the report and checks for the shared partition key. The major disadvantage of GRPEF is the need to localize the nodes since localization involves complex mathematical computation. It is also prone to selective forwarding attack. Context-Aware Architecture for Probabilistic Voting-based Filtering Scheme [29] has been proposed in such a way that it can be integrated with existing systems to enhance the given system. This scheme has three architectural modules: a filtering scheme with keying function, communication subsystem (responsible for collecting data), and Context-Aware Architecture (CAA) module. CAA is accountable for transforming sensor data into spatial-temporal data. CAA analyzes the data and identifies the compromised node. But the whole process drains the energy resource of the node which makes the scheme weak.

Commutative Cipher-based En-route Filtering (CCEF) is a cipher-based en-route filtering scheme in which each sensor node has a unique ID and a secret key [12]. The nodes detect their location and forward the position to the sink. Sink uses this information for verification. Sink initiates the query per session and prepares two keys (session key and witness key) to ensure that the query is initiated and the response is received securely. The major limitation of this scheme is the low en-route filtering efficiency. An Energy-aware Routing and Filtering Node (ERF) Selection in CCEF has been proposed for extending the lifetime of the network [13]. Key dissemination is done only to the intermediate nodes, and the path is created by considering distance and energy after key dissemination. Neighbouring nodes around the CH participate in report generation after CH receives the query from sink. CH forwards the final report to the corresponding intermediate nodes, which holds the disseminated key. Source has to know the routes and the authentication keys of all the nodes, and this makes the system weak. Adaptive En-Route Filtering to Extend Network Lifetime in Wireless Sensor Networks (AEF) is an energy-aware filtering scheme based on a fitness function [30]. It focuses mainly on energy savings with a minimal number of nodes in a cluster. Predeployed keys are assigned for every different path to address dynamic query. However, key independence is not achieved in this work. An En-Route Scheme of Filtering False Data (AEF) is a one-way key function [31]. Each sensor node is assigned a one-way key chain during deployment. An initialization message is sent to the sink through the chosen path by CH. A node in the path stores the value from the message randomly. CH forwards the report to the sink after receiving the report from the detecting nodes. En-router verifies the report while transmitting based on the information shared with detecting nodes. The need for reinitiation of the keys for energy session makes the scheme feeble.

Yang et al. proposed a scheme which uses primitive polynomial [18]. Authentication polynomial and check polynomial are derived from the primitive polynomial, and these polynomials are stored in node memory. The keys derived from the assigned polynomials are referred to as Message Authentication Polynomials (MAPs). MAPs are used for generating and verifying the report. En-routers receive the report and wait for warning messages. En-router checks the report only if it does not receive the warning message. The disadvantage of this scheme is the key-deriving procedure. An attacker is likely to launch by selective forwarding attack in such scenario.

Table 1 summarises the significance of related works and its limitations. We have observed that many of these schemes have two major limitations: the en-routers will not identify a compromised source node and the compromised en-routers

Table 1: Pros and cons of various filtering schemes.

S. no.	Schemes	Significance	Pros	Cons
1	SEF	(i) Nonoverlapping key partition (ii) Bloom filter for reducing overhead	(i) Supports dynamic topology (ii) Easy to apply	(i) T threshold limitation (ii) Does not work with the compromised partition
2	IHA	(i) Hop-by-hop authentication (ii) Pairwise key establishment	(i) Works on node failure (ii) Path-based filtering	(i) Maintenance cost is high (ii) Path dependency
3	STEF	(i) Query-based approach (ii) Verification based on tickets	(i) Verify the report validity (ii) Only ticket holders forward the report	(i) The unnecessary dropping of reports due to route failure
4	BECAN	(i) CNR-based authentication (ii) Bit-compressed scheme	(i) Key independency (ii) Ensures reliability	(i) Need prior knowledge about nodes in the path (ii) Communication overhead
5	CCEF	(i) Cipher-based authentication (ii) Secure query/response session	(i) Works for dynamic networks (ii) No threshold limitation	(i) Poor filtering (ii) Need extra care for report dissemination
6	CAEFS	(i) Integrated with all the other schemes to provide security (ii) Context-aware approach	(i) Compromised node isolation (ii) Resiliency	(i) Energy consumption is high
7	ERF	(i) Extension for CCEF (ii) Path creation based on distance	(i) Key dissemination only to an intermediate node (ii) Extends the lifetime of the nodes	(i) Need extra care for report dissemination (ii) CH requires prior knowledge of path and keys
8	MIHA	(i) Multipath authentication (ii) Keys are derived using a hash function	(i) Works for disjoint/braided path (ii) Route switching by the sink	(i) Node exploitation due to multiple paths (ii) Consumes more energy
9	DEF	(i) Uses hill climbing approach for key distribution (ii) Cluster-based approach	(i) Works independently on dissemination (ii) Suitable for dynamic topology	(i) Utilize more energy (ii) Report delays
10	LEDS	(i) Location-based filtering scheme (ii) A symmetric key approach for filtering	(i) No threshold limitation (ii) Provides E2E security	(i) Need location-aware key (ii) Relies on a path for report forwarding
11	GRPEF	(i) Location-based filtering scheme (ii) Key derivation using a multiaxis approach	(i) Supports sink mobility (ii) No threshold limitation	(i) Localization is complex (ii) Utilize more energy
12	AEF	(i) Based on a fitness function (ii) Provides security with a minimal number of nodes	(i) Energy-aware scheme (ii) Address dynamic query	(i) No key independency
13	NFFS	(i) Position-based filtering scheme (ii) Decides whether the report generating nodes are logical	(i) No threshold limitation (ii) Provides energy efficiency	(i) Not for dynamic networks
14	PCREF	(i) Uses MAP (ii) Cluster-based approach	(i) Does not have a fixed path (ii) Cluster-based approach	(i) T threshold limitation (ii) Storage overhead
15	KAEF	(i) One-way authentication (ii) Verification based on stored information	(i) New keys for every session (ii) The hash function for chain maintenance	(i) Need for key reinitiation
16	TICK	(i) Time-based filtering scheme (ii) Key generation based on clock function	(i) No need for a key exchange	(i) Not for an uncontrolled environment

go unnoticed/undetected. Reliability and End-to-End (E2E) delivery are questionable since the compromised nodes may drop the reports about an event due to which a sink cannot react. Hence, we propose an effective verification scheme termed as RRFS to overcome the limitations as mentioned earlier.

3. System Model

The system model refers to the set of abstractions of the functional behavior of a system. The system model defined or assumed has a profound impact in simulation and analysis of the system. To simulate, analyze, and evaluate the scheme, we have included models for two entities, namely network and threat which are explained in Sections 3.1 and 3.2, respectively. To make our paper easier to understand and comprehend, we have used a set of symbols and abbreviations, as shown in Table 2.

3.1. Network Model. WSN are usually deployed in an unattended environment. We assume that the density of the nodes in the WSN is high, and they do not move. Every node has a unique identifier (ID). Every sensor node has a well-defined transmission range. Nodes are assumed to be bidirectional. The nodes reside in the nearby location of the event are grouped to form a cluster, and one among the node is chosen as the cluster head (CH). The role of CH is circulated among the members of the cluster to avoid depleting the energy available with a particular node. CH is selected dynamically based on the residual energy. Nodes are not allowed to join multiple groups in the same session. Clusters and CHs are dynamic based on the requirement. One of the major assumptions is that all nodes immediately after deployment remain uncompromised since it happens in a systematic and controlled way. The sink is assumed to have a sufficient amount of resource in terms of computing, communication, and energy. Sink has the information of keys and can detect the clusters which are responsible for report generation. The sink is trustworthy, and the decisions will be taken based on the sink's reaction. Figure 1 shows the scenario of cluster formation and forwarding of reports to the sink. Brown color represents sensing nodes, black color represents cluster head, and green color is the sink. The colored pentagon represents sensing clusters, and the regular pentagon represents forwarding clusters.

3.2. Threat Model. Nodes are secure up to the connection establishment. An adversary can compromise nodes only after detecting events in an environment, which means that nodes can be compromised only when communication between nodes begins in wireless mode. In addition, we assume that an adversary can compromise all the nodes in the network, including CH in the cluster. We further believe that one among the multiple clusters forwards the genuine reports to the sink. Once a node is compromised, the attacker can access all its keying materials that are currently stored in the node.

Table 2: Terms, abbreviations, and symbols.

Terminology	Description
RRFS	Rank-based Report Filtering Scheme
G_k	Guest key
CH	Cluster head
S_k	Selfish key
CS	Cluster Score
ECDH	Elliptical curve Diffie-Hellmen
ECDLP	Elliptic curve discrete logarithm problem
P	Private key
Q	Public key
S	Score
ECC	Elliptical Curve Cryptography
DN	Downstream node
UN	Upstream node
MAC	Message Authentication Code
IR	Intracluster Report
ICR	Intercluster Report
RI	Reliability Index
ECH	En-routing cluster head
H	Honesty
K	Kindness
St	Strength
PR	Private Report
ECM	En-routing cluster member
CM	Cluster member
DHE	Diffie-Hellmen
Ts	Timestamp
T	Threshold energy
E	Event area
s	Total number of nodes
k	Number of nodes in cluster
m	Number of compromised nodes in a cluster
w	Number of compromised nodes in forwarding area
l	Number of CHs in forwarding area
e	Number of compromised CHs in forwarding area
d	Total number of nodes in "E"
l_d	Number of CHs in "E"
f	Number of compromised nodes in "E"
e_d	Number of compromised CHs in "E"

4. Preliminaries

4.1. Suitability of ECC for WSN Security. Strength of any cryptosystem depends on the size of the key, the larger the key, the more secure the scheme is, and it does not rely on the encryption algorithm used. However, in WSNs, this principle cannot be applied since the longer keys result in more processing overhead and thereby leading to quick depletion of the energy available with the nodes. Fortunately, Elliptical Curve Cryptography (ECC) provides an effective solution in protecting the network from the adversaries and intruders [32]. ECC is able to offer the same level of security or even better security when compared to other security schemes like RSA and DSA. Many security schemes based on RSA and DSA employ a key size of 1024 bits whereas a key size of 160 bits is sufficient in ECC to provide the same level of security. Another reason why ECC performs well in WSNs is that the intruders can solve the discrete logarithmic

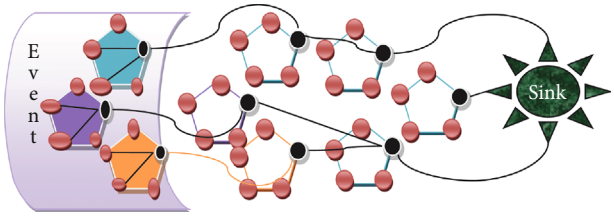

Figure 1: Formation of cluster and forwarding of reports to the sink.

problem (DLP) associated with other schemes in subexponential time. In contrast, it requires exponential time for the intruders to solve the associated DLP with ECC [33]. Moreover, the key generation process of ECC ensures that the newly entering malicious nodes is not able to find the generator points easily. In the context of cryptography, an elliptic curve is a plane curve represented by equation (1).

$$y^2 = x^3 + ax + b, \tag{1}$$

where x and y are the coordinates and a and b are real numbers and for the curve to be nonsingular, the discriminant $\delta \equiv -16(4a^3 + 27b^2)$ should not be equal to zero. Let p denote a large prime number and let F_p be the finite field of p elements then the elliptical curve $y^2 = x^3 + ax + b$ over F_p is represented as $E_p(a, b)$. In nonsingular EC, group operations are defined at point O as follows:

(i) Existence of identity: $(P + O = O + P)$.

(ii) Existence of inverse: $P_1 = (x_1, y_1)$, then $P_1 + P_2 = O$, where $P_2 = (x_1, y_1), P_1, P_1 \in E(F_p)$ which is an inverse of P_1 $(-P_1)$

(iii) Additive operation: (if $P_1, P_2 \in E(F_p)$, then $P_1 + P_2 = P_3 \in E(F_p)$, where $P_3 = (x_3, y_3)$; $x_3 = (\text{slope})^2 - x_1 - x_2$, $y_3 = \text{slope}(x_1 - x_3) - y_1$

(iv) Doubling operation: (if $P_1 \neq (-P_1)$; $2P = (x_3, y_3)$, $x_3 = \text{slope}^2 - x_1 - x_2$, $y_3 = \text{slope}(x_1 - x_3) - y_1$.

Slope is represented in equation (2).

$$\text{slope} = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & : P_1 + P_2 \\ \dfrac{3x_1^2 + a}{2y_1} & : 2P. \end{cases} \tag{2}$$

4.2. Elliptical Curve Diffie-Hellmen. Elliptical Curve Diffie-Hellmen (ECDH) is the secure key exchange algorithm in a nonsecure channel [16]. It is efficient than Diffie-Hellmen-Ephemeral (DHE). The general DH equation is as in equation (3).

$$k = g^{ef} \bmod p. \tag{3}$$

In the above equation, e and f values are randomly chosen and fixed for further calculation. But in DHE, e and f values are chosen during the runtime. DH/DHE uses modular arithmetic to compute the secret keys. ECDH is the advanced version which uses the elliptical curve to generate keys. The significance of ECDH lowers the computational power, storage, and memory usage. It achieves perfect forward secrecy by using temporary key pairs for each sensor. The key pairs are generated afresh for each session, and they are discarded from Random Access Memory (RAM) once the session expires. It is challenging to solve ECDLP even if the attacker obtains the private key. However, ECDH does not provide authentication. Hence, we are going for an authentication scheme after sharing the key securely. The authentication scheme is required to check the integrity of messages and is used in our research work as a variant of Message Authentication Code (MAC).

4.3. Message Authentication Code. A Message Authentication Code (MAC) is used to check the integrity of the message and assures that the message is from the intended sender [34]. Hence, this small piece of information is able to ensure the integrity and authenticity of the message. It consists of three phases, namely key generation, signing, and verification. Many MAC schemes are available in the literature, and most of them suffer from high computational overhead, lack of scalability, and lack of resilience to attacks. Among various types of MAC schemes, hash-based MAC (HMAC) is widespread and probably more secure and can be implemented using standard cryptographic primitives. Other available MAC schemes are Cipher Block Chaining MAC (CBC-MAC) and MAC based on universal hashing (UMAC). MACs can be created either using symmetric key-based approaches [35] or public key-based approaches [36]. Among various symmetric key-based approaches, the secret key polynomial based message authentication scheme is considered the most effective one. However, when the number of messages transmitted exceeds a threshold, the polynomial can be fully recovered, which makes the cryptosystem weak. The alternative is the public key-based approach, and the drawback with this asymmetric approach is high computational overhead [37]. However, recent surveys show that ECC-based public key schemes have desirable features such as low computational complexity, low memory storage, and high security resilience. In order to provide high-level protection, the hash function should be one-way and collision-resistant. Let $h(.)$ be a cryptographic hash function where $h : E(F_q) \rightarrow Z_n$ and $E(F_q)$ is an elliptic curve on the finite field. A MAC of keyed hash is defined as in equation (4).

$$\text{MAC}(m, k) = h(m \| k) \bmod q, \tag{4}$$

where m, k, and q denote message, key, and order, respectively.

5. Proposed RRFS

The RRFS scheme proposed in this research work composes of six major phases, namely predeployment, postdeployment, event processing, Reliability Index (RI), Cluster Score (CS), and report verification. These phases are described in Sections 5.1, 5.2, 5.3, 5.4, 5.5, and 5.6, respectively.

5.1. Predeployment. The sink chooses Elliptical Curve Cryptography for secure communication [38]. For efficient ECC implementation, sink chooses a finite field over $q(F_p)$, and the field representation is normal. Pseudo-Mersenne prime (p) is used in EC as reduction modulo. A cyclic subgroup with prime order "q" is formed using $G(q.G = O)$. $E(F_q)$ is a set of points satisfying equation (1), where $p > 3$. In nonsingular EC, group field operations can be defined over F_p. The projective coordinate is used here, which has three coordinates $(P_1 = (X_1, Y_1, Z_1))$ with $Z_1 \neq 0$. Usually, points are defined in affine coordinates $P_1 = (x_1, y_1)$. Mapping is required for converting affine to projective coordinate. Forward mapping of $P_1(x_1, y_1)$ is $P_1 = (x_1 Z_1, y_1 Z_1, Z_1)$ and reverse mapping of P_1 is $(X_1/Z_1, Y_1/Z_1)$.

$$\text{slope} = \begin{cases} \dfrac{Y_1 Z_2 - Y_2 Z_1}{X_1 Z_2 - X_2 Z_1} & : P_1 + P_2 \\ \dfrac{3X_1^2 + aZ_1^2}{2Y_1 Z_1} & : 2P. \end{cases} \tag{5}$$

For point addition, we have to perform a projective form test $(X_1/Z_1 = X_2/Z_2)$. By cross-multiplication, we get $(X_1 Z_2 = X_2 Z_1)$ and apply these values in equation (2) for $P_1 + P_2$. Slope for the point operation in projective coordinate is given in equation (5). The sink chooses domain parameters $\{E(F_p), n, G(x_1, x_2), h, q\}$, where $h = 1$. Sink installs domain parameter for all the nodes in the network. All nodes in the network should have a key pair: private key (P_i) and public key (Q_i). P_i and Q_i are distinct for each node. All other parameters are common to all nodes. Sink selects a scalar value "P_i" and does multiplication with "$G(x_1, y_1)$" to get "Q_i." Algorithm 1 shows the predeployment algorithm performed by the sink.

5.2. Postdeployment. The postdeployment phase has two activities, namely the connection establishment and guest key generation. These two activities are described in Sections 5.2.1 and 5.2.2.

5.2.1. Connection Establishment. After deployment, all the nodes are referred to as orphans. Communication takes place between nodes only when nodes are connected. The connection between report generating nodes and forwarding nodes should be made for forwarding the event details. Connection establishment begins with the sink by initiating beacon (relation) signals. The relationship signal is recursively sent to all nodes. After this signal has been received, the node will detect the ID of the nodes that are $T + 1$ hops away, where T is the threshold for the number of nodes to validate the report [22]. Figure 2 shows how the connection is made between the nodes. The relationship between the same color nodes shows that these nodes are connected and resides on the same path. The difference between the upstream (UN) and the downstream (DN) node forms a relationship.

$$\text{UN} - \text{DN} = t + 1. \tag{6}$$

Equation (6) checks the connectivity of the nodes. After the connection establishment, two nodes share a key between them is represented as guest key (G_k), and G_k changes for every session. Reports are verified between the nodes using G_k. Report verification takes place only if the two nodes are connected. Reports are verified by UN using G_k. Algorithm 2 explains the steps involved in the process of establishing a link between the nodes.

5.2.2. Guest Key Generation. After deployment and link setup, the connected nodes N_i share G_k and is kept secret for every session. G_k is recalculated for every session to improve security. For G_k calculation, connected nodes need Q of the connector. The process for generating a unique guest key is shown in Figure 3. G_k is calculated based on ECDH [16]. Nodes have a domain parameter in common. Private keys are picked randomly from the interval $(1, q - 1)$. Each node calculates and publishes the Q_i. When nodes are connected, downstream node (DN) initiates generation of G_k by multiplying P_{DN} with a public key (Q_{UN}) of its upstream node (UN). Guest key generation is based on scalar multiplication. UN also performs the same calculation for its $G_k(P_{\text{UN}} \times Q_{\text{DN}})$. For preventing node compromise, public keys are generated dynamically to prevent the attackers from compromising the connected nodes in the network, and this is shared only between the connected nodes. Even though the attacker is able to obtain the key resides in the current node, it is useless for the next session. Attacker cannot acquire the details of the connected node because of key independency. Algorithm 3 describes the steps taken to generate the guest key.

5.3. Event Processing. The activities that take place immediately after the occurrence of an event include dynamic group formation and report generation. These two activities are explained in Sections 5.3.1 and 5.3.2.

5.3.1. Dynamic Group Formation. Nodes sensing an event form multiple clusters with k members in each cluster, and one of the cluster nodes acts as a cluster header (CH). Nodes that have higher energy in the vicinity of the event are trying to become CH by sending RREQ to their one-hop neighbours [39]. After receiving RREQ, node accepts the request by sending the RREP to the node to which it is willing to join as a group member. After receiving RREP from "k" neighbours, CH sends ACK signal to the upstream node, and the group is closed after selecting "k" neighbours. Here, the threshold of "k" is restricted to six. After group formation, CH ignores RREQ packets from another cluster member. Cluster changes for every session. Another condition that imposed on the nodes is that a node cannot join multiple groups. Factors required for joining the group do not come under the purview of our research work.

CH is responsible for forwarding/receiving the reports to/from the en-routing group. After cluster formation, the key established within the group is called as selfish key (S_k). The CH of the current session is responsible for ensuring that the nonmembers of the group will not get access to this S_k. It is computed dynamically so that every group has a unique S_k

Require:
$N = N_1, N_2, \cdots, N_n$ is the set of "n" nodes present in the network except the sink
$D = (E(F_p), q, G(x_1, x_2), p)$//domain parameters
E_{init}: initial energy
ENSURE: Install (P_i, G_i) in N_i
for each node $N_i \in N$ do
    Load E_{init}//Energy required for operation
    Load a D//includes G, p, q, F_p
    Select P_i where $1 < P_i < p - 1$//Private key
    Compute public key $Q_i(x_i, y_i) = P_i.G(x_1, x_2)$
    Install (P_i, G_i) in N_i
end For

Algorithm 1: Predeployment algorithm.

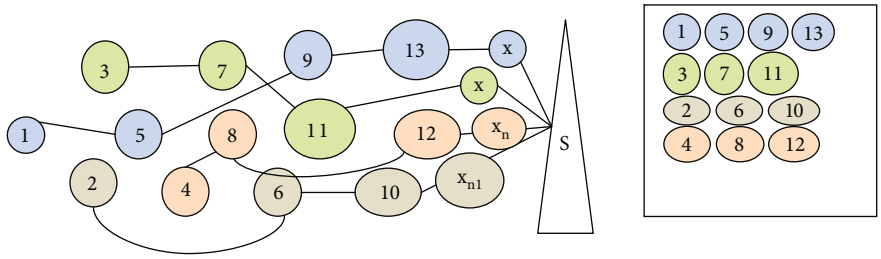

Figure 2: Connection establishment.

//SINK: the base station or the data collection unit
SN: the set of all nodes in the sensor network-SINK N_i is ith node
ID (N_i): identifier of N_i
d_i: distance between N_i and the sink S
k: no. of nodes to which the sink broadcasts the relation signal
M_i is the ith node among k neighbours to which the relation signal is sent by the sink
$\text{RS} = S_N - M_1, M_2, \cdots, M_k$//set of nodes in the network except the nodes to which the relation signal is sent by the sink
Condition to be met: $(\text{ID}(M_1), \text{ID}(M_2), \cdots, \text{ID}(M_k)) > \text{ID}$ (any node in RS)
T: the threshold for the number of nodes to validate the report
//index i refers to a neighbour of the sink
FOR $i = 1$ to k do
    relation-propagate (n_i)
//Function used to establish the families in the network.
//index j refers to a neighbour of neighbour i
relation_propagate (n_j)
neighb_set = neighbours (n_j)
//if statement refers to the condition that must be met to terminate the family establishment process
if (neighb_set = Φ) or $(\text{ID}(n_k) < t + 1))$
        terminate relation_propagate
else
        for every node x in neighb_set
        if $|(\text{ID}(x) - \text{ID}(\text{neighb}(x))| == t + 1$
            establish relation between $\text{ID}(x)$ and $\text{ID}(\text{neighb}(x))$
        end if
        relation_propagate (x)
    end for
end if

Algorithm 2: Connection establishment algorithm.

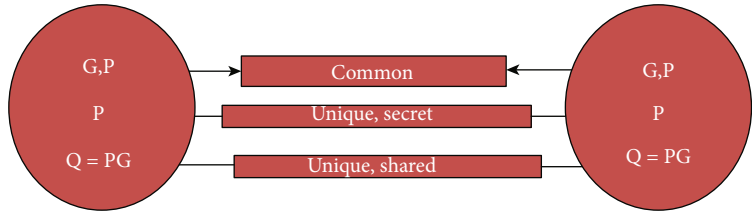

Figure 3: Guest key generation process.

SN: the set of all nodes in the sensor network-SINK
$M = MN_1, MN_2, \cdots, MN_K$//neighbour of the sink
RN = SN − M and [ID(MN_1), ID(MN_2), ⋯, IDMN_K)]>[ID(of any node in RN)]
DN and UN are the corresponding downstream node and upstream node.
DN sends the initiate signal to the UN in RN
for $i = 1$ to k do
  compute public key Q_i
  for $i = 0$ to n do
    calculate $Q = P$, G (add "G" "P" times)
  end for
  for $j = 1$ to m
    DN multiplies its private key with the public key of UN $P_{DN} \times Q_{UN}$ where Q_{UN} is $(P_{UN} \times G) \times P_{DN}$
    UN multiplies its private key with the public key of DN $P_{UN} \times Q_{DN}$ where Q_{DN} is $(P_{DN} \times G) \times P_{UN}$
    The product computed by DN and UN is equal
    "x" coordinate of this product is the shared secret G_k between DN and UN
    $G_k = x$
    end for
  end for

Algorithm 3: Guest key generation.

for each session. It plays a major role in calculating the Reliability Index (RI) for the nodes. RI computation is discussed in Section 5.4.

CH validates the member nodes by asking a random value from "k" neighbours [40]. The member nodes send random number which is represented as "L_i" where $i = 1, 2, \cdots, k$ to the CH. After receiving "k" different values of "L," the CH computes the S_k by bitwise XORing the received "k " values of "L" as in equation (7).

$$S_k = L_1 \oplus L_2 \oplus + .. \cdots L_k. \tag{7}$$

CH broadcasts S_k to the "k" neighbours for further communication within the cluster. Group members can communicate among themselves using S_k. S_k helps in calculating honesty (H) among member nodes by determining whether any neighbouring node in the group is compromised or not by comparing the reports exchanged between them. S_k is valid only within the group. Algorithm 4 depicts group formation.

5.3.2. Report Generation. After an event occurs, it should be detected, and a report about an event should be generated and forwarded to the sink. CH is responsible for forwarding the message in the form of a tuple <SENS_DATA, Ts> where SENS_DATA indicates the sensed data and Ts stands for the timestamp. Members after checking the consistency of the tuple generate three types of reports: Private Report (which is later verified by the sink), Intercluster Report (verified by the en-routers), and Intracluster Report (verification within a cluster). The node generates a Private Report (PR) with P. Intercluster Reports (ICR) are generated using G_k, and they are forwarded to the sink via the en-routing clusters. The cluster member generate intracluster reports (IR) using the S_k. These reports are verified within the cluster by the fellow member nodes, and their scope is confined to the cluster. Figure 4 shows the types of reports generated by the nodes in the network.

(1) Forwarding of Private Report (PR). Once the CH broadcasts the tuple <SENS_ DATA, Ts> to all other nodes in the cluster, CMs check whether the tuple is consistent or not. If the tuple is consistent, PR about an event are generated by the detecting nodes for sink verification. Details about an event is encrypted with "P," and forwards the reports to the CH. Equation (8) shows the MAC for PR.

$$\text{MAC}(m, P) = h(m||\text{Ts}, P), \tag{8}$$

$$\text{PR} = \{m, \text{Ts}, \text{MAC}\}. \tag{9}$$

SN: the set of all nodes in the sensor network-SINK
$M = MN_1, MN_2, \cdots, MN_K$
CH sends the RREQ signal to "k" neighbours
F_i: the ith family in the network
M_i is the ith node among k neighbours to which CH broadcasts
for $i = 1$ to k do
    RREQ-propagate (n_i)
    relation_propagate (n_j)
    neighb_set = neighbours (n_j)
    Node x joins the group by sending ACK to corresponding CH
    if $|(K| <= t + 1$
        CH sends RREP
        establish relationship between n and CH
    end if
end for
Let m be the cluster member
L-number assigned for the path of "m" member
$S_k \leftarrow L_1 // S_k$ is the selfish key of the cluster
for $j = 2$ to k
    $S_k \leftarrow S_k \oplus L_j$
end for

Algorithm 4: Group formation.

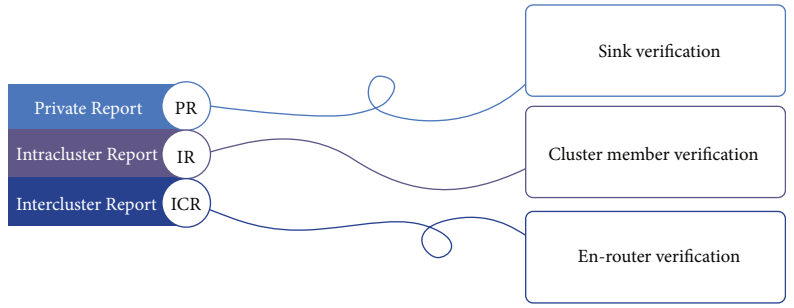

Figure 4: Types of report.

CH receives the report from CM, and XOR then reports and the forwards the reports to sink. PR format is shown in equation (9). CH forwards $\mathrm{PR} = (\mathrm{PR1} \oplus_1 \mathrm{PR2} \oplus_2 \mathrm{PR3} \oplus_3 \mathrm{PR4} \oplus_4 \mathrm{PR5} \oplus_5 \mathrm{PR_{CH}})$ to sink through its upstream-connected nodes. PR is verified only by the sink.

(2) Forwarding of Intracluster Report (IR). For proving authenticity within the cluster, CMs generate MAC along with the timestamp Ts. A node in the cluster encrypts the message about an event by S_k. Let $h(.)$ be a cryptographic hash function, and MAC is computed as in equation (10) using the S_k.

$$\mathrm{MAC}(m, S_k) = h(m \| \mathrm{Ts}, S_k). \tag{10}$$

Once the MAC is computed, a node sends the IR as shown in equation (11) to other members of the cluster.

$$\mathrm{IR} = \mathrm{Ts}, \mathrm{MAC}. \tag{11}$$

Upon receiving an IR, a member node verifies the report by computing its MAC and compares it with the received MAC. A match or mistake between these two MACs plays a role in the computation of Reliability Index (RI), since the Reliability Index depends on the honesty parameter.

(3) Broadcasting of Intercluster Report (ICR). Cluster member generates Intercluster Report (ICR) for en-router verification. Generation of ICR is also very similar to the way in which IR is generated. However, how MAC is computed differs between these two types of reports. The difference noted is the use of G_k in MAC computation. Message Authentication Code (MAC) is generated by the downstream node for en-router verification is shown in equation (12), and equation (13) shows the format of ICR.

$$\mathrm{MAC}(m, G_K) = h(m \| \mathrm{Ts}, G_K) \bmod q, \tag{12}$$

$$\mathrm{ICR} = m, \mathrm{Ts}, \mathrm{MAC}. \tag{13}$$

CM forwards ICR to CH for further verification in en-routing phase. CH gets IR and ICR from all CMs and checks the MAC of IR for RI calculation. CH forwards the reports to multiple ECH. En-routing group CH is referred to as en-routing CH (ECH). Algorithm 5 explains the procedure followed to create the three types of reports.

5.4. Reliability Index Computation and Intracluster Verification. Reliability Index (RI) is a metric that is computed to determine the extent to which the node can be trusted in the transmission of the event report to the sink. This helps, in particular, the en-routing cluster to find out whether or not the downstream clusters are reliable. The trust established between the en-routing nodes is referred to as entrust. Entrust is of two types: direct en-trust between two directly connected nodes and third party entrust is determined through an intermediate node. Upstream nodes thwart node compromise attacks and false data injection attacks based on RI.

Honesty (H), strength (St), and kindness (K) are the three parameters of RI that represent the genuineness, available energy, and the overhead of the number of packets forwarded in a time interval δ_t, respectively, in a figurative sense. For each node, these three parameters are assigned either 0 or 1 by the other $k-1$ nodes in the group. The time interval δ_t is defined as $\delta_t = t_i + 1 - t_i$. The honesty of the nodes is calculated based on the intrareports (IR). IR is generated by encrypting the event details with the selfish key (S_k). After receiving the IR, fellow members and CH begin to calculate H by verifying the received report. CM receives IR from all members and checks whether the already received mac_i and the $\bar{\text{mac}}_i$ (which is computed using S_k) are the same or not by XORing these two. If both mac_i and $\bar{\text{mac}}_i$ are equal, then $\bar{\text{mac}}_i \oplus \text{mac}_i$ results in zero; otherwise, $\bar{\text{mac}}_i \oplus \text{mac}_{ij}$ results in a nonzero value. If the report is true, a value of 1 is assigned to H, otherwise "0" is assigned. The honesty of a node ($H_i(t)$) is expressed as shown in equation (14).

$$H_i(t) = \begin{cases} 1 & : \text{MAC}_i = \text{MAC}_j \\ 0 & : \text{MAC}_i \neq \text{MAC}_j \end{cases} \tag{14}$$

Let us assume that a node sends k reports in the interval δ_t. Then, honesty score is computed as in equation (15).

$$H_i(t) = \sum_{j=1}^{k} H_i(j). \tag{15}$$

Strength (St) is nothing but the energy of the nodes. Determining the strength predicts whether the node is able to accomplish the forwarding of reports and data verification. CH calculates threshold energy (T) of the nodes based on the average energy level of the nodes in the cluster as in equation (16). Let us represent the residual energy be E_{resid}, and the threshold energy be T.

$$T = \sum_{i=1}^{k} \frac{E_{\text{resid}}}{k}, \tag{16}$$

where k stands for the number of nodes in the cluster, including the CH. If the node's E_{resid} is equal to or above the threshold level, the St is calculated as "1." Let the number of control packets and the number of data packets transmitted by a node in the interval δ_t be NC and ND, respectively. Then, the total number of packets transmitted by that node in the interval δ_t is defined as in equation (17). St is computed by consistency in forwarding both control packets and data packets.

$$T_p = N_C + N_D \tag{17}$$

The value for strength of node i in the interval δ_t is computed as in equation (18).

$$\text{St}_i = \sum_{j=1}^{T_p} S_i(j). \tag{18}$$

St helps in calculating the E_{resid} of the nodes. St_i of a node i is assigned either 1 or 0 after E_{resid} is shown in equation (19).

$$\text{St}_i(t) = \begin{cases} 1 & : \text{if } E_{\text{resid}} >= T \\ 0 & : \text{if } E_{\text{resid}} < T \end{cases} \tag{19}$$

The kindness (K) is decided only on data packets. It is computed in terms of the dropping rate, i.e., the ratio between the number of packets received by the node and the number of packets dropped by a particular node. A node initiates a timer and broadcasts the packet to nodes. If that node does not receive the packet before time out then, K is assigned a value of 0, otherwise 1. The kindness (K) value in the interval δ_t is computed as in equation (20).

$$K_i(P) = \begin{cases} 1 & : K = 1 (\text{if packet } P \text{ is forwarded}) \\ 0 & : K = 0 (\text{if packet } P \text{ is not forwarded}). \end{cases} \tag{20}$$

Let the number of packets received by node i in interval δ_t be NP. The kindness score is computed as in equation (21).

$$K_i(\text{NP}) = \sum_{i=1}^{\text{NP}} K_i(P). \tag{21}$$

The Reliability Index of node i is computed as the average of these three parameters is shown in equation (22).

$$\text{RI}_i = (H_i + S_i + K_i)/3 \tag{22}$$

5.5. Cluster Score Computation. After checking the IR, CM starts to calculate the RI against all the nodes in its group. The RI which ranges from 0 to 1 (malicious to trustworthy) is stated in equation (25). For example, CM_1 calculates the RI for all members based on equation (22) and updates the

REQUIRE: CH: cluster head
CM – $\{CM_1, CM_2, \cdots, CM_k\}$: the set of cluster members
ENSURE: return IC
Step 1: CH broadcasts the message in the form of a tuple <SENS_DATA, Ts> to all elements of CM
Step 2: CM receives the tuple
Step 3: If <SENS DATA, Ts> is consistent//Private Report
CM calculates PR_{CM} using P_{CM}, forwards it to connected CH
CH calculates IR using S_K//Intracluster Report
$IR_{ip} = \{m, Ts, MAC\}$
CM calculates ICR using G_k //Intercluster Report
$MAC_{DN} = MAC(m \| Ts, G_k)$
$ICR = \{m, T, MAC\}$
Else
CH changes and go to Step 1.
Step 4: End if
Step 5: return $ICR = (ICR_1, \cdots, ICR_k)$

Algorithm 5: Report generation.

RI value to CH. CH receives $k-1$ "RI" values for each node. Now CH calculates the score for all the cluster member. Score (S) of a node is the average of RI that are given by the fellow member, including CH. Score calculation is done by CH as given in equation (23).

$$S = \frac{1}{k}\sum_{i=1}^{k} RI_i. \tag{23}$$

After score calculation, CH calculates the Cluster Score (CS) for the cluster. CS is the average of the score of all the nodes in the group. It is stated in equation (24).

$$CS = \frac{1}{k}\sum_{i=1}^{k} S_i. \tag{24}$$

After calculating the metrics, nodes/clusters are identified as in equation (25).

$$CS(x) = \begin{cases} 0 < x <= 0.5 & (\text{node is compromised}) \\ 0.5 < x <= 1 & (\text{node is not compromised}) \end{cases}. \tag{25}$$

5.6. Report Verification. ECH receives the report from downstream CH. ECH verification takes place only if the nodes are connected. Sink also verifies the report that escapes the en-router verification. En-router verification and sink verification are explained in Sections 5.6.1 and 5.6.2. Intracluster verification is already explained in Section 5.4.

5.6.1. Intercluster Verification. After receiving ICR from downstream CH, ECH checks the integrity of the message and Ts. If it fails, ECH asks for CS and informs sink about CS. If the report is consistent, ECH notifies the group for message arrival by broadcasting the message and the report. En-routing cluster member (ECM) receives the reports and looks for its downstream connection node. If connected, ECM checks whether the already computed mac_i and this $\bar{mac}_i$ are one and the same or not by XORing these two. If both mac_i and $\bar{mac}_i$ are equal, then $\bar{mac}_i \oplus mac_i$ results in zero; otherwise, $\bar{mac}_i \oplus mac_i$ results in a nonzero value. If $\bar{mac}_i \oplus mac_i$ results in zero, it indicates that the report received from the downstream node is valid and generates its own ICR for the next en-routing group. If the verification fails, CM drops the report and informs CH. ECH checks the score of corresponding DN. If the score is low, ECH marks DN as malicious. If the score is high, ECH marks CM as malicious. CH maintains a score table. It holds the score of all the nodes in cluster. After verification, ECH forwards ICR to its upstream-connected node for next verification. ICR sometimes is forwarded without any verification if the downstream nodes are not connected. Sink will do the verification for the report that the en-routing group has missed. Figure 5 shows the process of ICR verification. Algorithm 6 explains the steps of ICR verification

5.6.2. Sink Verification. Sink receives the report from the detecting nodes which is encrypted using private key (P_i). Sink is able to decrypt the report and get the message about an event. It waits for the details from the en-routers to take necessary action against the event. Sink receives the ICR from the multiple clusters that are just one hop away from the sink. Sink verifies PR by computing ($\bar{mac}_i \oplus mac_i$). If report matches, sink verifies ICR received via the en-routing cluster head (ECHs). Sink compares both the messages (received through PR and ICR). If both the messages are the same, the necessary action will be taken against the event without any further delay. If there is a mismatch among messages, sink looks for CS and query the CH in E which have high CS and initiate an action based on the corresponding CH. Irrespective of multiple verifications that takes place in the network, there is a possibility for few reports go undetected in intermediate stages, and these reports are called escaped

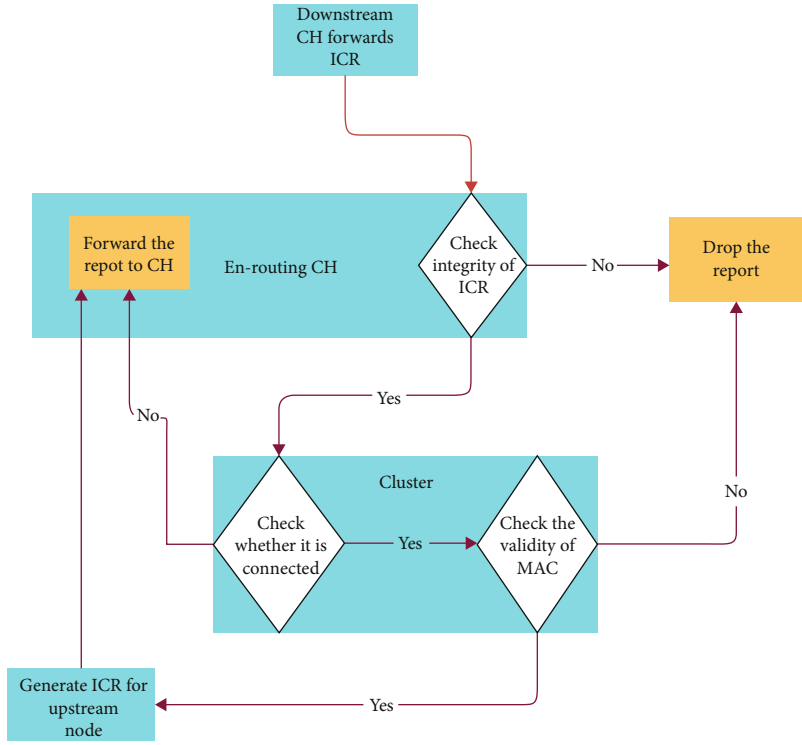

Figure 5: ICR verification.

REQUIRE: $ICR = [ICR_1, \cdots, ICR_k]$, $IDs_{DN} = [N_1, \cdots, N_k]$, $CM = [CM_1, \cdots, CM_k]$
OUTPUT: ICR for en-routers, True, False
Step 1: CH checks the integrity using T
Step 2: CH broadcasts ICR, ID_{DN} to CM
Step 3: UN_i checks whether it can verify the report by checking ID of DN
Step 4: Set Value = True
Step 5: If (UN – DN == $t + 1$)//Check whether the downstream node is connected
If ($\bar{mac}_{UN} \oplus mac_{DN} \neq 0$)//verifying MAC
CM set Value = false//drop the report and informs CH
Else
$mac_{DN} = MAC(m, G_k)$
ICR = mac_{DN}//Compute ICR for upstream verification
Forward ICR to CH//Next hop verification
Else
Forward ICR to CH//no downstream connected nodes.
Return value
End if
Step 6: Return value

Algorithm 6: ICR verification.

reports. However in our scheme, the sink is able to identify these escaped reports and verify them. If there is a mismatch, sink collects the S from CH and exclude compromised nodes from participating in the next session and reinitiate the relation signal. Thus, sink filtered the false data that are injected through the en-routers.

6. Performance Evaluation

6.1. Filtering Analysis. In RRFS, nodes sensing the event organize themselves as a cluster and report about the event to the sink. Let us assume that sensing nodes in the detecting area are represented as "d" in "E." Clusters are formed with "k" nodes (which are also described as neighbour nodes of CH or cluster members). The number of member nodes in a cluster is restricted to 6, i.e., $k \leq 6$ for active filtering. "l_d" is the total number of cluster heads (CH) in "E." Normal nodes are represented as "k." Malicious nodes that reside in "E" is denoted as "f."

6.1.1. Scenario 1. Attacker compromises "m" among "k" nodes in a cluster, i.e., all compromised nodes are normal

nodes, and CH is trustworthy. Hence, the probability mass function is stated as in equation (26).

$$P_r(X) = \frac{\binom{k}{m}\binom{d-k}{f-m}}{\binom{d}{f}}. \quad (26)$$

6.1.2. Scenario 2. Adversary compromises "e_d" number of CHs (among "l_d" CHs) and "m" nodes (among "k" nodes) in "E." The probability of injecting false data in region "E" is stated in equation (27).

$$\mathrm{FP}_E = \sum_{m=i}^{k} \frac{\binom{k}{m}\binom{d-k}{(f-e_d)-m}}{\binom{d}{f-e_d}} \cdot \sum_{e_d=j}^{l_d} \frac{\binom{l_d}{e_d}\binom{d-l_d}{(f-m)-e_d}}{\binom{d}{f-m}}, \quad (27)$$

where "l_d" is the total number of cluster heads (CH) in "E" and "e_d" is the compromised CH among l_d. FP_E is the success probability of injecting false data within "E." Probability of filtering injected false data in "E" is $\mathrm{FP}_A(1 - \mathrm{FP}_E)$. FP_A gives the filtering probability of RRFS in area "E." Figure 6(a) shows the success probability (FP_A) of filtering false data with compromised nodes in "E." RRFS effectively filters the false data within the cluster in the area "E" even with the increased number of compromised nodes in the cluster "E." In Figure 6(a), the SEF filtering rate decreases considerably with the increase in the false report because of compromised key pool. BECAN does not withstand as the compromised nodes in the cluster increases beyond the threshold. RRFS does not have any threshold limitation, since members share selfish key within the cluster, and it can verify the report of the cluster members and successfully filtered the reports. PF_A reaches "1" theoretically, which is not possible in real time. RRFS is simulated using ns-3 with 1000 nodes deployed. Fifty nodes are sensing the event with 10 CH in "E." Each cluster has 3 to 6 nodes. Simulations conducted by us include ten random events and 6-14 en-routers. For each event, simulations were run 100 times. FP_A is evaluated as a ratio of a total number of false reports filtered to the total number of false reports injected into the system which is represented as in equation (28).

$$\mathrm{FP}_A = \frac{\text{Number of filtered false data}}{\text{total number of false data}}. \quad (28)$$

Figure 6(b) shows the experimental results of FP_A. FP_A of RRFS increases with an increase number of compromised nodes when comparing with the existing schemes. This implies that RRFS fulfil our design. We are distributing the attackers in the area "E." Eventually, it shows that there is a drop in the filtering probability when more than 20 nodes are compromised in the area. It implies that some amount of false data in the cluster remain unnoticed, and it successfully passed the verification. RRFS efficiently handles the filtering when compared to other schemes.

After verifying the reports in the sensing area E, we are evaluating how RRFS works while forwarding the reports to the sink (en-router verification). RRFS forwards ICR to sink through en-routers. Let "H_M" be the maximum number of hops to reach the sink. Sometimes, false data forwards H hops attacker can inject false data even after IR verification. Let us assume that the total number of nodes in the forwarding area is "s." Among "s," "w" nodes are compromised. Attacker has to compromise "m" nodes in the cluster to get high score, since en-router verification relies on RI. Hence, the success probability (P_{FC}) of injecting false data through forwarding cluster is represented as in equation (29). Probability ($\mathrm{FP}_{\mathrm{FC}}$) of filtering false data through en-routers is stated as in equation (30).

$$P_{\mathrm{FC}} = \frac{\binom{k}{m}\binom{s-k}{w-m}}{\binom{s}{w}}, \quad (29)$$

$$FP_{\mathrm{FC}} = 1 - P_{\mathrm{FC}}. \quad (30)$$

Figure 7(a) shows the probability ($\mathrm{FP}_{\mathrm{FC}}$) of filtering false data through en-routers for the various filtering scheme. F P_{FC} is the ratio of filtering false data to the injected data. Filtering probability ($\mathrm{FP}_{\mathrm{FC}}$) increases when the number of en-routers increases. The reason behind the increase in F P_{FC} is due to the fact that the false data is filtered with the increase number of hops. Figure 7(b) shows the simulated results of the proposed scheme in terms of the number of hops, i.e., how report filtering varies as the number of hops increases. 1000 nodes are deployed with the transmission range of 15 m in the region $100 \times 100\ \mathrm{m}^2$. Each cluster has 3 to 6 nodes. We are experimenting with 10 random events, and we vary the en-routing nodes from 6 to 14 for a single event. The number of times the simulation was executed is 60, and the average value is plotted. We are injecting false data through en-routers. RRFS achieves high filtering ratio even with the increase in number of compromised nodes in the cluster. This is because node goes through two-tier verification using IR and ICR. Upstream nodes successfully filtered the false data of the corresponding downstream nodes. $\mathrm{FP}_{\mathrm{FC}}$ is not linear in Figure 7(b) because the verification takes place across the network. Filtering probability decreases at some point which indicates that the false data escapes the en-route filtering. RRFS deviates only by 10% from the ideally expected error rate. All other schemes suffer from more than 30% deviation from the ideal error rate.

6.2. False Acceptance Ratio Analysis. For injecting phoney reports, the attacker has to acquire at least ($r = k/2$) cluster member to get decent score. By acquiring S_k, attackers inject false data to the en-routers and also get approval from the fellow members, since it is a cluster-based verification. After getting S, malicious node generates report for the upstream-connected nodes. We are considering forwarding area here.

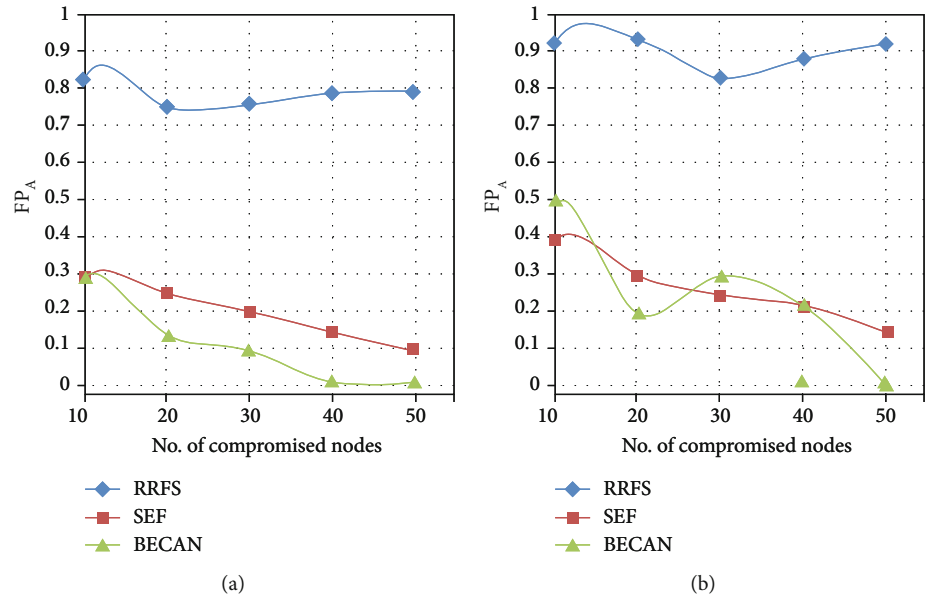

FIGURE 6: Shows the probability (FPA) of filtering false data within the cluster in "*E*." (a) Compromised node vs. filtering probability (theoretical). (b) Compromised node vs, filtering probability (simulation).

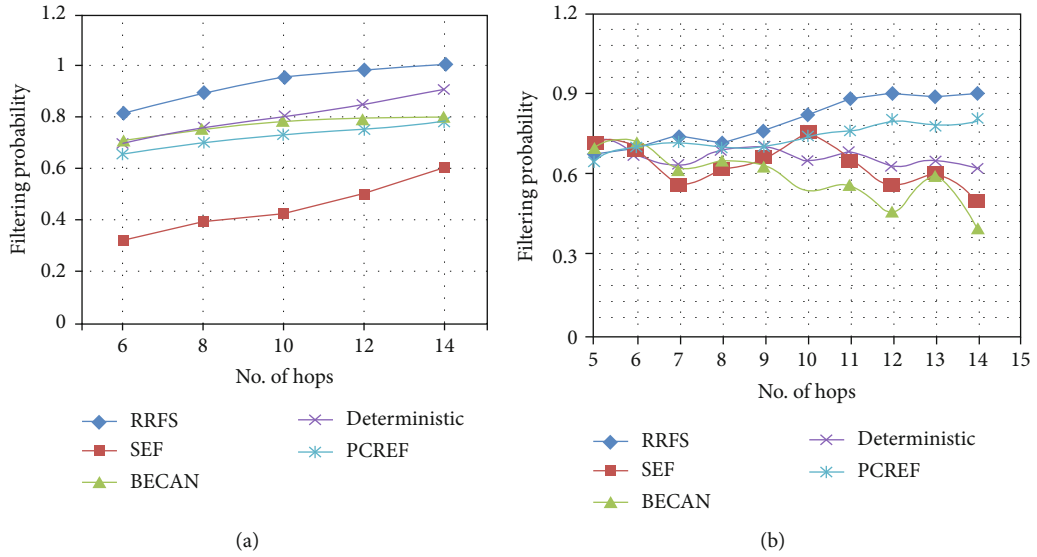

FIGURE 7: The probability of filtering through en-routers. (a) Filtering through en-router (theoretical). (b) Filtering through en-router (simulation).

Among s nodes, w nodes are compromised. Each cluster has k nodes. Hence, the success probability for gaining access to a cluster is represented as $P_m = \sum_{c=1}^{r} P_{\mathrm{FC}}$. It is defined as the ratio of proving their genuiness. Figure 8 shows the simulation results of injecting false data into the network. It is also defined as the ratio between the total number of false reports to the total number of false reports injected into the network, and it is represented as False Acceptance Ratio (FAR), which is represented in equation (31). We are compromising m nodes in the cluster. We are clustering the nodes as 6, 8, 10, 15, and 20. FNR decreases when the cluster has 6 members. It represents that limiting the cluster size works well with the proposed scheme. Cluster with 6 and 10 nodes also drops the false data. FAR is low when the cluster has 15 members. FAR increases slowly when the number of nodes in the cluster increases, which implies that the number of compromised node in the cluster increases ($k = 20$). Hence, it is difficult for the scheme to filter the false data. By decreasing the nodes in the group, it is possible to reduce the deployment cost, and also, we can filter the reports with minimum hops. RRFS even works well with the increase in the cluster size ($k > 6$).

$$\mathrm{FAR} = \frac{\text{Number of false data accepted}}{\text{total number of false data}}. \tag{31}$$

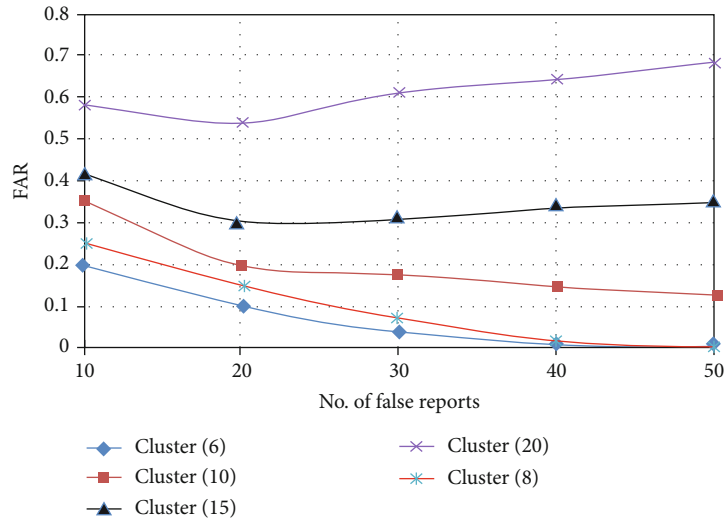

Figure 8: FAR.

6.3. Detection of Phoney Reports. In the proposed scheme, CH collects the MAC from members. A CM encrypts the report with S_k and forwards it to the CH. CH receives a report of 8 bytes. The cost of sending/receiving a byte is 11.76 μ J/17.89 μJ, with a transmission rate of 38.4 kbps, respectively, in MICAz mote. A node takes 1.67 ms to transmit IR to CH. The CM also forwards ICR for en-router verification. Hence, an ICR takes 5 ms to reach the CH. Total transmission time (TT_R) is the time it takes to reach a sink from the detection area, and it is calculated as in equation (32).

$$TT_R = CT \times L \times H \tag{32}$$

CT is the time taken for transmitting the report within the hop, L is the length of the report (MAC), and H_M is the maximum number of hop to reach the sink. L is of 24 bytes. Hence, a report takes 75.6 ms to reach the sink. Based on our simulation results, the execution time for session key is 2863 ms (which includes precomputation). Time delay for detecting the next event is represented as TD_E, which is stated in equation (33).

$$TD_E = T_s + T_f + T_v. \tag{33}$$

T_s is the time needed for session key generation, T_f is the time taken for report forwarding, and T_v is the time used for verification. By substituting the values (T_s, T_f, T_v) in equation (33), we will get the TD_E. Time delay for next event detection is 2.89 s.

6.4. Energy Consumption. We discuss the energy consumption of RRFS in terms of computation and communication, and this helps in analyzing the lifetime of WSN [11]. RRFS scheme has employed ECC for improved security with less overhead. Among the various possible curves, we have chosen the 160-bit curve with the reduction modulo of q (pseudo-Mersenne prime), since it provides adequate security and does not introduce excess computational overhead. A point on the EC (elliptic curve) is represented using the projective coordinate system that removes modular inversion, which is very expensive. ECC mainly depends on scalar multiplication. ECC point operation has been done using mixed point multiplication and repeated doubling. We have used a private key of 20 bytes in length, a public key of 40 bytes. Table 3 gives the simulation parameters for the implementation of the proposed scheme using ns-3. The initiation of cluster formation is based on the residual energy of the nodes. Sensing nodes which are having higher residual energy initiate the cluster formation. The consumed energy of a node is calculated as in equation (34)

$$E_{cons} = E_{comp} + E_{comm}, \tag{34}$$

where E_{comp} and E_{comm} are the energies spent in computing and communication, respectively. Let the energy of a node be E_{init}, and the residual strength of a node is computed as in equation (35).

$$E_{resid} = E_{init} + E_{cons}. \tag{35}$$

6.4.1. Energy Consumption for Computation. Energy consumption of RRFS mainly lies on session key establishment. The guest key is generated for every session in the network, which is represented as E_{G_k}. RRFS calculates a new session key for every session. The computational cost is calculated based on equation (36).

$$C = U \times I \times t. \tag{36}$$

U represents voltage, V represents the current drawn by a node when it is in active mode, and t is the execution time of the operation. We are using MICAz mote parameters to achieve a more realistic simulation. Based on simulation results, it has been observed that the execution time for G_k

Table 3: Simulation parameters.

Deployment parameters	Values
Area	200 × 200
Number of nodes	1000
No. of normal nodes	999
Sink	1
Antenna	Omni directional antenna
Radio propagation model	Two-way ground
Network interface type	Wireless Phy/MICAZ
Transmission range	>10 m
MAC type	802.15.4 MAC
Node to sense	Any
Node to receive	Sink
Node initial energy	100 J
Average neighbours	≤6

is 1028 ms. MICAz consumes 8 mA when it is in the current draw mode (mode with the radio off) and consumes 21.3 mA/14 mA for transmission/receiving. The voltage required for the operation of the mote is 3 V. The cost of the session key includes the calculation of public key (Q) (which includes selecting private key (P) and multiplication with "G" which is predeployed). It requires 1839 ms for executing the computation of keys. The cost calculation of E_{G_k} in RRFS is computed based on equation (37). Node requires 44.14 mJ for "Q" computation. Publishing of "Q" requires 0.17 mJ. The total cost of calculating and publishing "Q" is 44.31 mJ. The cost of the session key includes the computation of G_k at both the nodes. The computation cost of G_k is 24.67 mJ. The total cost of sharing the session key (CS_{G_k}) is defined in equation (37).

$$\mathrm{CS}_{G_k} = 2 \times C. \tag{37}$$

Hence, the computation of CS_{G_k} consumes 49.34 μJ. In other schemes, noninteractive keys without precomputaion (calculation of public keys) costs 55 mJ which is higher than our scheme. Each node has to compute the session key for all the nodes in the routing path, which in turn consumes a hefty amount of energy for key establishment. If there are five en-routers to the sink, the source node needs 275 mJ (5 × 55) for generating a noninteractive session key. In RRFS, G_k is between the connected nodes, and it costs 49.34 mJ. The cost of E_{S_k} computation of is 0.4 μJ. The energy spent on computation for a single session is represented as in equation (38).

$$E_{\mathrm{comp}} = E_{S_k} + E_{G_k}. \tag{38}$$

6.4.2. Energy Consumption for Communication. Let "p" be the probability of filtering the false data in an en-router. In RRFS, the probability of filtering false data within "h" hops is P_h as in equation (39).

$$P_h = 1 - (1 - p)^h. \tag{39}$$

We already discussed the filtering probability of RRFS. Let hm be the no. of hop false data travels. Then, the average no of hops false data travels within H in RRFS is given in equation (40).

$$h' = E(h) = \sum_{i=1}^{H} i.P_{h_i} \tag{40}$$

H is the maximum number of hop false data travels before it reaches the sink. Let us assume an adversary compromises $m(<k)$ nodes among k (cluster members), then the probability is $\sum_{c=0}^{m} P_{\mathrm{FC}}$, and the probability of a node being a CH is $1 \div 2$. The possibility of filtering the false data in an en-router is $p = \Pr.(1 - 1/2(m/k.s/w))$, where Pr is the probability of nodes being connected to DN. Probability ($P_{\mathrm{h_m}}$) of false data filtered after hm hops in RRFS is stated in equation (41).

$$P_{\mathrm{h_m}} = \sum_{c=0}^{m} P_{\mathrm{FC}}.(1 - p)^{h_m - 1} \times p. \tag{41}$$

Adversary can compromise m nodes in the cluster, and it is less than $r(m - 1)$. Adversary compromises ECH to inject false data. Hence, the probability of false data forwarded to the maximum number of hops in RRFS is given as P_H in equation (42).

$$P_H = \sum_{c=m}^{k} P_{\mathrm{FC}} + \sum_{c=0}^{m-1} P_{\mathrm{FC}} \times (1 - p)^{H-1} \tag{42}$$

Let L_m be of 24 bytes without any additional field. The length of the ICR is denoted as $L_{\mathrm{IR}}(L_m + (k \times L_{id}) + t)$. Total energy consumed for sending and receiving a byte is represented as E_T, and let HM be the number of hops to the sink. The energy consumption involved in transmitting a report without RRFS is shown in equation (43).

$$E_{\mathrm{WO}} = L_m.E_T.H_M(1 + \beta), \tag{43}$$

where 1 indicates the legitimate traffic, β represents the falsely injected data, and $10 > = \beta > = 1$. With RRFS, false reports are forwarded to h hops as in equation (40). The average energy consumption of filtering the false data with RRFS (E_{WR}) is given in equation (44).

$$E_{\mathrm{WR}} = L_{\mathrm{IR}}.E_{\mathrm{T}}.H_M\left(1 + h'\beta\right). \tag{44}$$

The overall communication includes both transmission and computation. Equation (45) gives the total energy consumption for RRFS (E_{RRFS}).

$$E_{\mathrm{RRFS}} = E_{\mathrm{comp}} + E_{\mathrm{WR}}. \tag{45}$$

The overall consumption includes both transmission and

computation. Equation (45) gives the total energy consumption for RRFS.

(1) Overhead Analysis. Nodes are preinstalled with keys and deployed in the environment. Every node has its own ID, upstream-connected node ID, S_k, P, Q, and G_k which is of 10 bits, 10 bits, 80 bits, 160 bits, 320 bits, and 160 bits. Every node in the RRFS needs 720 bits, and this consumes meager space in node's memory. Hence, the memory overhead is manageable in RRFS. For every session, nodes within the cluster send a number of r bits to CH. CH receives $r(k-1)$ bits and broadcasts S_k (r bits) within the cluster. Message overhead of CH and CM during S_k generation is $2r(k-1)$ bits and $2r$ bits, respectively. Nodes within the cluster send $r(k-1)$ IR in which r bits are present in each report. Hence, the message overhead of IR for CM within the cluster is 2 $r(k-1)$ bits. CM send/receive ICR of "l" bits to/from CH for en-router verification. Hence, the overall message overhead for CM within the cluster is $2r + 2r(k-1) + 2l$ bits. CH receives $r(k-1)$ IR reports from CM. CH also receives $l(k-1)$ from each CM. Hence, the message overhead for CH within the cluster is $2r(k-1) + r(k-1) + l(k-1)$ bits. Computational complexity of RRFS includes IR, ICR, and G_k computation. Every node in the cluster computes and verifies IR. Complexity of IR is $2\Omega(mr)$ where $m = \log N$ (length of the message) and $r = \log N$ (length of IR). Computation of G_k requires $O(N)$. Computation of ICR requires $2\Omega(ml)$ where $m = \log N$ (length of the message) and $l = \log N$ (length of IR). Computational complexity of RRFS is $O(N) + 2\Omega(m(r+l))$.

We include the energy spent in computation also while calculating the energy required for communication. We are simulating the environment with MICAz mote. The data rate of MICAz is 38.4 kbps with a current requirement of 3 mA. Moreover, MICAz takes 0.28 ms for transmitting a byte. Energy consumed for sending/receiving a byte is 11.76 μ J/17.89 μJ, respectively. ET is 29.65 μJ (11.76μJ + 17.89 μJ). Energy consumption of the proposed scheme is shown in Figure 9 to the amount of injected data with the true data within number of hops. $E(10)$ and $E(20)$ are the energy consumption of 10 and 20 compromised nodes which includes RRFS. It shows that E_{WO} increases to the end, and $E_{WR}(10)$ and $E_{WO}(20)$ are not linear since RRFS is filtering the false data with the average number of hops.

Figure 10 shows the experimental results of the proposed scheme with the comparison of various schemes. We are randomly generating events with the compromised nodes. The filtering efficiency is evaluated based on the proportion of legitimate reports, and the results are shown in terms of energy consumption. It is shown that average energy consumption decreases with the increase in the number of en-routers in RRFS. In RRFS, en-router verifies all the corresponding downstream node if they are connected. For successful verification, r nodes should be connected to DN nodes. ECH can detect node compromise attack using S (whether DN or CM is compromised). Pr should be in the desired range to filter the data within h hops. Pr is calculated as a ratio to the member nodes with the connected DN. Pr should be less than (DN/HM). In Figure 10, low energy consumption in subsequent hops implies high filtering efficiency of RRFS. False reports are filtered within h hops. In Figure 10, [18] consumes high energy because of deriving polynomial, and it has to store the authentication polynomial for future verification, whereas RRFS does not have computation like this. PKAEF [27] consumes lesser energy than [18], but because of a longer report, it consumes more energy than RRFS and AEF [32]; filtering capacity is low when compared to other schemes.

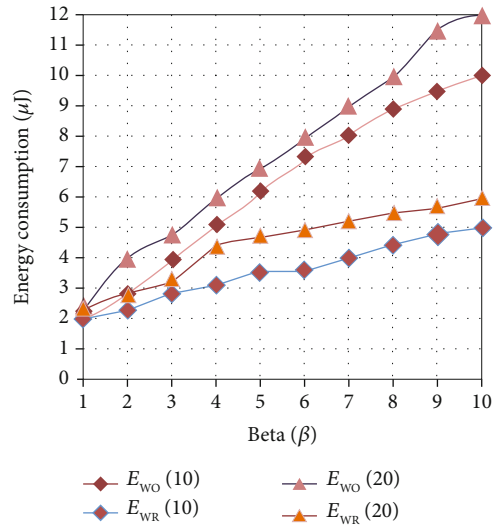

FIGURE 9: The energy consumption with respect to β (ratio of true data concerning false data).

Figure 11 shows false-positive rate (FPR = 100 × {FP/(FP + TN)}) vs. energy. We are randomly compromising nodes in forwarding area. FPR is high when the number of en-routing node is minimal. Figure 11 shows energy consumption is low when compared to other schemes for FPR. This is because RRFS groups the node into clusters with k neighbours. Hence, en-routing node increases in the forwarding area. This, in turn, helps the node to save energy, and also, false reports are filtered within minimal number of hops. RRFS consumes less energy compared to other scheme, which implies false data filtered with increase in number of nodes in the network. Proposed scheme works efficiently in detecting the false data with less energy consumption.

7. Discussion

Here, we analyze the capability of the RRFS scheme in sustaining the following four major attacks that are frequently launched in WSNs: (i) node compromise attack, (ii) false data injection attack, (iii) report disruption attack, and (iv) selective forwarding attack. We have found that the RRFS scheme is able to withstand all of the above attacks, and hence, we claim that RRFS is efficient. Table 4 summarises the filtering schemes and how they are sustaining in the different types of attacks that are launched in WSN.

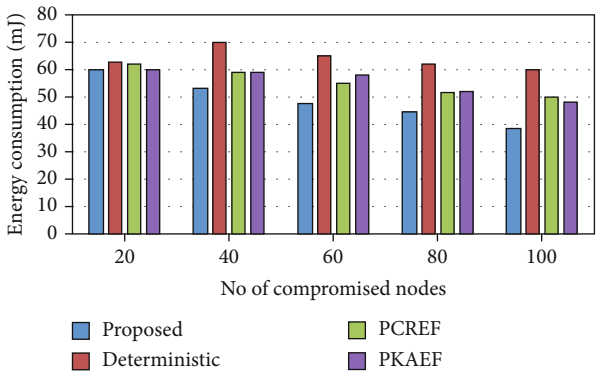

Figure 10: Energy consumption of various schemes based on filtering efficiency.

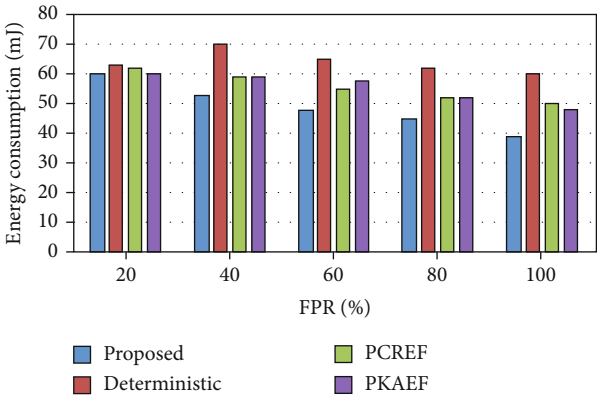

Figure 11: Energy consumption of various schemes based on FPR.

7.1. Node Compromise Attack

7.1.1. Statement. RRFS scheme is proficient in detecting node compromise attack.

7.1.2. Justification. Attackers capture the decisive materials (keys that are used for sending data) and compromise the node. Compromised nodes start sending a report on behalf of the adversaries. This is called node compromise attack [19]. For launching node compromise attack, attackers have to get access to S_k of a particular cluster before generating the report. An adversary can launch this attack while the reports are getting forwarded. ECH receives ICR and verifies integrity by computing ICR with G_k. If there is a mismatch, upstream-connected node asks for the score to confirm the attack. ECH will ask for node's S, which is connected in the downstream group. S is calculated based on RI. If the score of suspicious nodes is much less than 0.5, the corresponding node is declared as malicious and drops the reports. The ECH repeats this process for all the involved nodes in the downstream cluster. In case the S is high for all members in the group, ECH concludes that the CM is malicious and informs the sink about it. The RRFS scheme is efficient in detecting the compromised node irrespective of the node being either a cluster member or a cluster head. In DSDEA [41], CH holds the authority, and hence, once CH is compromised, the sensing domain can be easily falsified. RRFS CH only forwards the report and does not hold any authority. BECAN [24] is able to detect that node compromise has taken place. However, it is not able to identify which particular node has been compromised, which is possible in RRFS. PCREF [18] follows the SEF [21], and in this, if the whole partition is compromised, false data will escape the verification.

7.2. False Data Injection Attack

7.2.1. Statement. RRFS scheme provides resiliency in false data injection attack.

7.2.2. Justification. To launch false data injection attack, the adversary has to compromise nodes in the cluster before the report is generated. Hence, RRFS prevents false data injection even if the attacker is able to compromise $k-1$ nodes in the same cluster. It is basically difficult for the attacker to compromise the whole cluster before report generation since the cluster formation is dynamic for every session. In other schemes, usually, if the CH is compromised, it can successfully inject the false data to the en-routers. Since we have enabled ECH to detect the false data with the help of downstream CM reports, the RRFS is able to filter the downstream injected false data by checking the node's S. Another possible attack is gang injection attack in which a set of nodes collaborate among themselves to launch a colliding attack. RRFS detects gang injection of false data using a parameter called CS, which indicates the reliability of a cluster as a whole. Low value of CS indicates that most of the nodes in the cluster are compromised. This in turn helps ECHs to detect the gang injection of false data. RRFS prevents the compromised CM from forwarding the reports. Even if a few malicious reports escape from this level of prevention, the en-routing CHs can detect and prevent these reports. BECAN [24] is not able to detect gang injection of false data. GFFS [17] detects only whether the nodes are legitimate for report forwarding/generating. DSDEA [41] is not able to detect false data injection since compromised CH goes undetected.

7.3. Report Disruption Attack

7.3.1. Statement. RRFS is more efficient in detecting report disruption attack.

7.3.2. Justification. Member nodes broadcast IR between themselves to prove the legitimacy of clusters. Compromised member node in a cluster produces a false MAC using S_k; nodes detect the malicious activity by checking the correctness of IR. Even if a non-CH-compromised report escapes the cluster, an upstream-connected node of a non-CH compromised node verifies the correctness of ICR (using G_k). If a CH is compromised, an upstream ECH checks the correctness of the MAC in the received reports. Attackers sometimes compromise the entire cluster. Hence, the reports escape all the verification. Sink can match and verify the reports about an event since it receives multiple reports of the same event from the multiple clusters. Sink takes appropriate actions irrespective of the location where false data injection takes place and the entity which launched the attack. Hence, the proposed scheme is able to detect the report

Table 4: Comparison of various schemes.

S. no. of schemes	Attacks			
	Node compromise attack	False data injection attack	Report disruption attack	Selective forwarding attack
SEF	Prone	Not completely secure	Prone	Prone
IHA	Prone	Secure	Prone	Prone
STEF	Prone	Prone	Prone	Not completely secure
BECAN	Prone	Secure	Prone	Secure
CCEF	Prone	Not completely secure	Prone	Prone
PDF	Secure	Secure	Prone	Prone
ERF	Prone	Not completely secure	Prone	Prone
MIHA	Secure	Secure	Secure	Prone
DEF	Prone	Prone	Secure	Secure
LEDS	Prone	Secure	Secure	Secure
GRSEF	Prone	Not completely secure	Prone	Prone
AEF	Prone	Not completely secure	Not completely secure	Prone
NFFS	Secure	Secure	Prone	Prone
PCREF	Secure	Not completely secure	Prone	Prone
RRFS	Secure	Secure	Secure	Secure

disruption attack. DSEA [41] does not resist a compromised CH from forwarding the report. PDF works only for the compromised CH report verification. BECAN [24] is not capable of detecting report disruption attack if two or more numbers of nodes are compromised, whereas in RRFS, score helps to detect the compromised nodes. ERF [13] is vulnerable to report disruption attack since it depends on the probabilistic path for report forwarding. RRFS forwards the report in multiple paths.

7.4. Selective Forwarding Attack

7.4.1. Statement. RRFS is efficient in handling selective forwarding attack.

7.4.2. Justification. RRFS scheme ensures efficient forwarding by sending the reports in multiple paths to ensure that at least a single report about an event reaches the sink. Compromised intermediate nodes drop all legitimate reports passing through them purposefully, and this intentional dropping is called selective forwarding attack [20]. Multiple source nodes which are detecting the same event forward the reports in multiple paths. Hence, even if a compromised CH is reluctant in not sending the report, the sink will receive the report from some other source nodes. Hence, the selective forwarding attack has been overcome by the proposed scheme. ERF [13] is a query-based approach, and the path is established in between the selected CH, and hence, CCEF is prone to selective forwarding attack. NFFS [24] is vulnerable to selective forwarding attack since a CH holds the authority for forwarding the reports even after getting compromised.

8. Conclusion and Future Enhancement

Developing verification schemes for filtering the false data injected either by the attackers or by the compromising nodes in WSNs is difficult due to the nature of communication and the unfriendly terrains in which they operate. RRFS proposed in this paper is able to overcome various difficulties of the existing schemes in filtering the injected false data. In the proposed scheme, the nodes are randomly deployed in the environment. Groups are formed in the sensing area for report generation. Scores are calculated for the individual nodes as well as clusters. This helps to filter the injected data as early as possible. En-routing group verification scheme helps to check all downstream nodes that are taking part in report forwarding. For every event, two reports are forwarded to the sink. PR is forwarded to the sink through CH. CH forwards the report through connected nodes. ICR is generated using the G_k between the connected nodes. En-router/sink drops the report whenever a mismatch occurs. Besides these two, one more report that includes the MAC was generated using the selfish key, which is confined within the cluster. However, forwarding of IR is not through the en-routing groups. This scheme is able to deal with the situation where the CH itself is nonreliable. This scheme is able to thwart gang injection of false data. In future work, we will develop a scheme by predicting the node's behavior and recognizing the pattern of nodes by applying either Artificial Intelligence- (AI-) or Machine Learning- (ML-) based algorithms [42]. An effective scheme shall be designed in which the sink shall collect information about an event from specific clusters that are highly reliable. Another possibility of enhancing the work is to modify this scheme so that it shall be applied for the mobile wireless sensor networks [43].

Data Availability

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Ethical Approval

All procedures performed in studies involving human participants were in accordance with the ethical standards of the

institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

Consent

Informed consent was obtained from all individual participants included in the study.

Conflicts of Interest

The authors declare that they have no conflict of interest.

References

[1] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] G. Bovenzi, D. Ciuonzo, V. Persico, A. Pescapè, and P. S. Rossi, "IoT-enabled distributed detection of a nuclear radioactive source via generalized score tests," in *Advances in Signal Processing and Intelligent Recognition Systems. SIRS 2018*, vol. 968 of Communications in Computer and Information Science, pp. 77–91, Springer, Singapore.

[3] W. Lu, Y. Gong, X. Liu, J. Wu, and H. Peng, "Collaborative energy and information transfer in green wireless sensor networks for smart cities," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1585–1593, 2018.

[4] M. Jamshidi, E. Zangeneh, M. Esnaashari, A. M. Darwesh, and M. R. Meybodi, "A novel model of Sybil attack in cluster-based wireless sensor networks and propose a distributed algorithm to defend it," *Wireless Personal Communications*, vol. 105, no. 1, pp. 145–173, 2019.

[5] R. Singh, J. Singh, and R. Singh, "WRHT: a hybrid technique for detection of wormhole attack in wireless sensor networks," *Mobile Information Systems*, vol. 2016, 13 pages, 2016.

[6] X. Jinhui, T. Yang, Y. Feiyue, P. Leina, X. Juan, and H. Yao, "Intrusion detection system for hybrid dos attacks using energy trust in wireless sensor networks," *Procedia Computer Science*, vol. 131, pp. 1188–1195, 2018.

[7] H. Kalkha, H. Satori, and K. Satori, "Preventing black hole attack in wireless sensor network using HMM," *Procedia Computer Science*, vol. 148, pp. 552–561, 2019.

[8] Q. V. Do, T.-N.-K. Hoan, and I. Koo, "Optimal power allocation for energy-efficient data transmission against full-duplex active eavesdroppers in wireless sensor networks," *IEEE Sensors Journal*, vol. 19, no. 13, pp. 5333–5346, 2019.

[9] D. Ciuonzo, A. Aubry, and V. Carotenuto, "Rician MIMO channel- and jamming-aware decision fusion," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 3866–3880, 2017.

[10] Lidong Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.

[11] J. Yu, L. Feng, L. Jia, X. Gu, and D. Yu, "A local energy consumption prediction-based clustering protocol for wireless sensor networks," *Sensors*, vol. 14, no. 12, pp. 23017–23040, 2014.

[12] H. Yang and L. Songwu, "Commutative cipher based en-route filtering in wireless sensor networks," in *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, pp. 1223–1227, Los Angeles, CA, USA, 2004.

[13] M. K. Shahzad and T. H. Cho, "An energy-aware routing and filtering node (ERF) selection in CCEF to extend network lifetime in WSN," *IETE Journal of Research*, vol. 63, no. 3, pp. 368–380, 2017.

[14] T. P. Nghiem and T. H. Cho, "A multi-path interleaved hop-by-hop en-route filtering scheme in wireless sensor networks," *Computer Communications*, vol. 33, no. 10, pp. 1202–1209, 2010.

[15] Jianzhong Li, Lei Yu, Hong Gao, and Shuguang Xiong, "Grouping-enhanced resilient probabilistic en-route filtering of injected false data in WSNs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp. 881–889, 2012.

[16] R. R. Ahirwal and M. Ahke, "Elliptic curve diffie-hellman key exchange algorithm for securing hypertext information on wide area network," *International Journal of Computer Science and Information Technologies*, pp. 363–368, 2013.

[17] J. Wang, Z. Liu, S. Zhang, and X. Zhang, "Defending collaborative false data injection attacks in wireless sensor networks," *Information Sciences*, vol. 254, pp. 39–53, 2014.

[18] X. Y. Yang, J. Lin, W. Yu, P. M. Moulema, X. W. Fu, and W. Zhao, "A novel en-route filtering scheme against false data injection attacks in cyber-physical networked systems," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 4–18, 2015.

[19] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang, "Understanding node capture attacks in user authentication schemes for wireless sensor networks," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2020.

[20] D. C. Mehetre, S. E. Roslin, and S. J. Wagh, "Detection and prevention of black hole and selective forwarding attack in clustered WSN with active trust," *Cluster Computing*, vol. 22, no. S1, pp. 1313–1328, 2019.

[21] Fan Ye, H. Luo, Songwu Lu, and Lixia Zhang, "Statistical en-route filtering of injected false data in sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 839–850, 2005.

[22] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pp. 259–271, Berkeley, CA, USA, 2004.

[23] C. Kraub, M. Schneider, K. Bayarou, and C. Eckert, "STEF: a secure ticket-based en-route filtering scheme for wireless sensor networks," in *The Second International Conference on Availability, Reliability and Security (ARES'07)*, pp. 310–317, Vienna, Austria, 2007.

[24] Rongxing Lu, Xiaodong Lin, Haojin Zhu, Xiaohui Liang, and Xuemin Shen, "BECAN: a bandwidth-efficient cooperative authentication scheme for filtering injected false data in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 1, pp. 32–43, 2012.

[25] Zhen Yu and Yong Guan, "A dynamic en-route filtering scheme for data reporting in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 150–163, 2010.

[26] K. Ren, W. Lou, and Y. Zhang, "LEDS: providing location-aware end-to- end data security in wireless sensor networks," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pp. 1–12, Barcelona, Spain, 2006.

[27] C. Yi, G. Yang, H. Dai, L. Liu, and N. Li, "Public key-based authentication and en-route filtering scheme in wireless sensor networks," *Sensors*, vol. 18, no. 11, p. 3829, 2018.

[28] A. S. Uluagac, R. A. Beyah, and J. A. Copeland, "TIme-based DynamiC keying and en-route filtering (TICK) for wireless sensor networks," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Miami, FL, USA, 2010.

[29] S. M. Nam and T. H. Cho, "Context-aware architecture for probabilistic voting-based filtering scheme in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 16, pp. 2751–2763, 2017.

[30] M. K. Shahzad, S. M. Islam, K.-S. Kwak, and L. Nkenyereye, "AEF: adaptive en-route filtering to extend network lifetime in wireless sensor networks," *Sensors*, vol. 19, no. 18, p. 4036, 2019.

[31] T. Yuan, S. Zhang, Y. Zhong, and J. Ma, "KAEF: an en-route scheme of filtering false data in wireless sensor networks," *2008 IEEE International Performance, Computing and Communications Conference*, 2008, pp. 193–200, Austin, TX, USA, 2008.

[32] H. Arshad and M. Nikooghadam, "An efficient and secure authentication and key agreement scheme for session initiation protocol using ECC," *Multimedia Tools and Applications*, vol. 75, no. 1, pp. 181–197, 2016.

[33] S. Challa, A. K. Das, V. Odelu et al., "An efficient ECC-based provably secure three-factor user authentication and key agreement protocol for wireless healthcare sensor networks," *Computers and Electrical Engineering*, vol. 69, pp. 534–554, 2018.

[34] G. Tsudik, "Message authentication with one-way hash functions," *ACM SIGCOMM Computer Communication Review*, vol. 22, no. 5, pp. 29–38, 1992.

[35] D. Singh, B. Kumar, S. Singh, and S. Chand, "SMAC-AS: MAC based secure authentication scheme for wireless sensor network," *Wireless Personal Communications*, vol. 107, no. 2, pp. 1289–1308, 2019.

[36] Z. Huang, J. Lai, W. Chen, M. Raees-ul-Haq, and L. Jiang, "Practical public key encryption with selective opening security for receivers," *Information Sciences*, vol. 478, pp. 15–27, 2019.

[37] M. Wazid, A. K. Das, V. Bhat K, and A. V. Vasilakos, "LAM-CIoT: lightweight authentication mechanism in cloud-based IoT environment," *Journal of Network and Computer Application*, vol. 150, article 102496, 2020.

[38] Z. Jinchao, "Research on key predistribution scheme of wireless sensor networks," in *2012 Fifth International Conference on Intelligent Computation Technology and Automation*, pp. 280–290, Zhangjiajie, Hunan, China, 2012.

[39] H. S. U. Harn and C. F. Hsu, "Predistribution scheme for establishing group keys in wireless sensor networks," *IEEE Sensors Journal*, vol. 15, no. 9, pp. 5103–5108, 2015.

[40] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *2003 Symposium on Security and Privacy*, pp. 197–213, Berkeley, CA, USA, 2003.

[41] H. W. Ferng and N. M. Khoa, "On security of wireless sensor networks: a data authentication protocol using digital signature," *Wireless Networks*, vol. 23, no. 4, pp. 1113–1131, 2017.

[42] R. Bhatt, P. Maheshwary, P. Shukla, P. Shukla, M. Shrivastava, and S. Changlani, "Implementation of fruit fly optimization algorithm (FFOA) to escalate the attacking efficiency of node capture attack in wireless sensor networks (WSN)," *Computer Communications*, vol. 149, pp. 134–145, 2020.

[43] M. Wazid, A. K. Das, N. Kumar, A. V. Vasilakos, and J. J. P. C. Rodrigues, "Design and analysis of secure lightweight remote user authentication and key agreement scheme in internet of drones deployment," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3572–3584, 2019.