Resilient Aggregation in Simple Linear Sensor Networks

Kevin J. Henry · Douglas R. Stinson

Abstract A sensor network is a network comprised of many small, wireless, resource-limited nodes that sense data about their environment and report readings to a base station. One technique to conserve power in a sensor network is to aggregate sensor readings hop-by-hop as they travel towards a base station, thereby reducing the total number of messages required to collect each sensor reading. In an adversarial setting, the ability of a malicious node to alter this aggregate total must be limited. We present three aggregation protocols inspired by three natural key pre-distribution schemes for linear networks. Assuming no more than k consecutive nodes are malicious, each of these protocols limits the capability of a malicious node to altering the aggregate total by at most a single valid sensor reading. Additionally, our protocols are able to detect malicious behavior as it occurs, allowing the protocol to be aborted early, thereby conserving energy in the remaining nodes. A rigorous proof of security is also given for each protocol.

Keywords wireless sensor network · resilient aggregation · linear network · key pre-distribution

1 Introduction

A wireless sensor network is an ad-hoc network comprised of many computationally limited wireless sensor nodes whose job is to collect readings about their environment. These readings are periodically forwarded to one or more *sinks* or *base stations*, which are often assumed to be far more powerful than a typical sensor node. The severe resource constrains on sensor nodes makes many cryptographic tools, such as public-key cryptography, too expensive to rely on in many settings. Thus, protocols that rely solely on symmetric-key schemes are often preferable.

Martin and Paterson [17] have provided a framework which summarizes the various types of sensor networks, motivated by how symmetric keys are distributed among nodes. Their framework separates nodes into categories based on the expected topology of the network, the capabilities of the individual nodes, and what the expected patterns of communication are within the network. One important distinction is a *homogeneous* sensor network, where all nodes are identical, versus a *hierarchical* network, where there are nodes of varying capability. Hierarchical networks provide a natural tree structure with the capabilities of individual nodes increasing when moving from a leaf towards the root.

Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
Tel.: +1-519-888-4567
E-mail: {k2henry,dstinson}@uwaterloo.ca

In this paper we consider the problem of increasing network efficiency by aggregating sensor readings on a hop-by-hop basis for a special class of sensor networks, known as *linear sensor networks*, in the presence of an active adversary. As the name implies, a linear network topology contains many sensor nodes arranged along a straight line. This matches many natural deployment models for sensor networks, such the monitoring of subway tunnels, pipelines, or perimeter monitoring. Although several solutions have been proposed for secure aggregation in sensor networks, most of them assume a high node density or a hierarchical node structure, and are therefore poorly suited to linear networks.

Our goal is to investigate what assumptions are necessary to achieve resilient aggregation in linear sensor networks in the presence of an active adversary for a variety of linear network settings. Protocols for three different settings are given, based on three different symmetric key pre-distribution schemes. Each of these protocols limits the capability of an adversary to providing a single valid reading to an aggregate total, or terminates early upon detection of a maliciously modified reading.

1.1 Linear Sensor Networks

Linear networks are a natural deployment model for sensor networks. Applications such as pipeline, subway, border, or perimeter monitoring are inherently linear. Jawhar and Mohamed [14] give a classification of linear sensor networks that focuses on the topology of the network (thin, thick, or very thick), as well as a hierarchical classification of nodes according to capability and role within the network.

The simplest linear network is a *thin / one-level* network of uniformly distributed identical sensor nodes deployed along a straight line. Such a network may also be referred to as a *simple* linear network or a *one-dimensional* network.

Definition 1 A *simple linear network* is a connected non-cyclic graph where each node has exactly one neighbor, or two distinct neighbors. The two nodes with only one neighbor are referred to as the *endpoints* of the network.

Although the physical arrangement of nodes is linear, the fact that nodes communicate wirelessly means that a node may be able to communicate with more than just its direct neighbors in the physical network. Thus, the *communication graph* may have additional edges not present in the physical network graph.

Definition 2 A (N, d)-*linear network* is a linear network containing N nodes, each able to communicate with nodes up to d hops away. The nodes of such a network are denoted by $n_1, n_2, \ldots, n_N$ where nodes n_1 and n_N are the endpoints, and node n_i is located between nodes n_{i-1} and n_{i+1} for $1 < i < N$.

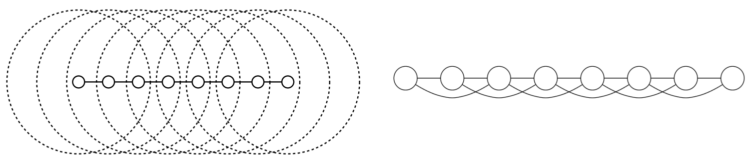

Fig. 1 An $(8, 2)$-linear sensor network. The left diagram shows the physical layout of the network along with dotted circles denoting the communication range of each node. The right diagram shows the communication graph for the same network

Fig. 1 shows an example of an $(8, 2)$-linear network.

We refer to nodes that are directly adjacent to each other as *neighbors*, and neighbors at distance d if there are $d - 1$ nodes in between them.

In terms of the framework given in [17], a simple linear network is:

- Homogeneous - All nodes that are not endpoints are identical.
- Fixed - Nodes are not mobile.
- Full control - Given the sequential deployment nature of linear networks, it is reasonable to assume that the order nodes are deployed in is fully controlled.

These properties, in particular, the high degree of control over the deployment of nodes, allow for the use of very efficient key pre-distribution schemes to facilitate secure communication.

An important factor that distinguishes linear sensor networks from more general networks is the expected difference in node density. In general, sensor networks are modeled as nodes deployed on a flat plane, with each node having a given communication radius. Because linear networks are one-dimensional, it is expected that far fewer nodes will be located within any given node's communication range. In the extreme case, each node in a linear network may have at most two neighbors within its communication range.

Although the linear problem setting differs from that of most general networks, linear sub-networks are often utilized in general networks, such as simply routing a single message between two nodes. Therefore, depending on the specific setting, protocols for linear networks can serve as building blocks for protocols in more general settings.

1.2 Aggregation

In general, the job of each sensor node is to collect and forward sensor readings to a base station. However, transmitting a message is extremely expensive in terms of energy use. In settings where an aggregate reading (e.g. the sum of all the individual readings of the nodes) is sufficient, an *aggregation protocol* can be used to collect aggregate sensor readings using significantly less network communication than if each node reported their results individually. At the most basic level, such protocols involve passing an aggregate sum hop-by-hop toward one of the endpoints of the network, with each node adding its individual reading into the aggregate sum before passing it on.

In an adversarial setting, a malicious node may:

- Report an invalid sensor reading.
- Arbitrarily alter the aggregate total.
- Deviate from the aggregation protocol (i.e., refuse to forward messages).

Our goal is to construct a *secure aggregation protocol* in which a malicious node can modify the aggregate total by at most a single valid sensor reading. That is, any attempt to otherwise alter the aggregate total can be detected by other nearby nodes. Note that unless additional assumptions are made about the distribution of nodes or the environment, such as the property that any given reading will be sensed and reported by at least two nodes, it is impossible to force a malicious node to report its true reading, as opposed to a valid possible reading.

Secure aggregation protocols that are concerned only with the integrity of the aggregate total are often referred to as *resilient data aggregation protocols* [25]. In some settings it may be desirable to not only ensure the integrity of the aggregate total, but also the secrecy of the aggregate total and of each individual node's sensor readings. Such aggregation protocols are referred to as *concealed data aggregation protocols (CDA)* [10] or *private data aggregation protocols (PDA)* [11] depending on the aggregation topology and the privacy goals of the protocol [6].

Like other sensor network topologies, a linear sensor network usually contains one or more distinguished nodes, known as base stations. A base station is often assumed to have more power and greater computational and communication abilities than a regular sensor node, and in the linear setting the base station will generally be one or both endpoints of the network. Our protocols will assume a single base station at one endpoint of the network.

1.3 Related Work

The general problem of aggregation in ad-hoc networks and sensor networks without any security considerations has been well studied. Fasolo et al. [8] provide a good survey of existing approaches. In terms of secure aggregation, protocols can generally be categorized into resilient, concealed, and private protocols.

Wagner [25] formally investigated which aggregation functions are possible to securely compute, demonstrating that in some settings even simple aggregate functions, such as the sum or average, are insecure without certain assumptions in place. More recently. Manulis and Schwenk [16] have provided a formal security model and framework for resilient aggregation, alongside a protocol that is secure in their model. Other examples of resilient aggregation protocols include SIA by Przydatek et al. [21] and improvements in [9], as well as protocols by Roy et al. [24] and Hu and Evans [13]. Each of these protocols relies on a hierarchical aggregation topology, and are therefore unsuited to a linear setting. We do note that one of our protocols is similar in structure to the approach in [13]; however, our assumptions about key pre-distribution and the means by which we achieve security are completely different. Other approaches to providing resilient aggregation include [4], which is based on detecting statistical anomalies with the assumption that nearby nodes are likely to have correlated readings, and [15], which provides multiple homomorphic MAC constructions based on both symmetric-key and public-key approaches.

Chan and Castelluccia [6] provide a formal security framework for concealed and private aggregation. Peter et al. [20] provides a survey of concealed data aggregation protocols, while Bista and Chang [3] provide a survey of private data aggregation protocols separated into three categories: perturbation-based, shuffling-based, and homomorphism-based. Perturbation-based approaches work by scrambling individual sensor readings before sending them to an aggregator who is able to recover the aggregate, but not individual readings. Examples of such approaches include CPDA [11] and PRDA [19]. Shuffling-based approaches break individual sensor readings into multiple pieces that are sent to multiple destinations. Examples include SMART [11] and improvements on it [26], and iPDA [12], with the latter also focusing on integrity/resiliency as well. Homomorphism-based approaches utilize homomorphic cryptosystems to allow the aggregation of encrypted data. Examples include Girao et al. [10], Armknecht et al. [1], and Castelluccia et al. [5].

1.4 Our Contributions

Although the problem of secure aggregation is well-studied, most approaches are hierarchical in nature, with each node forwarding readings to an aggregator node higher up in the tree. Non-tree based approaches generally rely on nodes to collaboratively select a clusterhead that performs aggregation for all other nearby nodes. The nature of linear networks makes these general approaches less useful, as traffic naturally flows in one direction towards the base station, and nodes are not expected to have very many neighbors. In some cases, such as the protocol presented in Sect. 3, nodes are only capable of communication with their two direct neighbors in the network. Hierarchical or cluster-based approaches provide no benefit in this setting. In general, by exploiting precise knowledge about both the topology and distribution of cryptographic keys, we are able to construct resilient aggregation protocols for simple linear networks that are both efficient, and allow the detection of malicious nodes locally within the network.

We present three protocols for resilient data aggregation built specifically for simple linear sensor networks, inspired by three natural key pre-distribution protocols for linear networks, as described in the next section. Although two of our protocols are similar to the approach in [13], we exploit the higher degree of control over key distribution in sensor networks to detect malicious node behavior as it occurs, rather than during a separate verification phase after the main protocol is finished. This allows the remaining nodes in

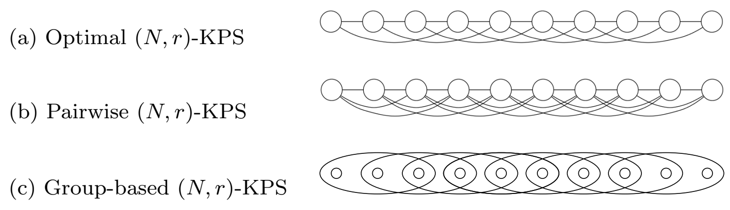

Fig. 2 Optimal and pairwise KPS for a $(10, 3)$-linear network, and group-based KPS for a $(10, 2)$-linear network. In (a) and (b), shared keys are denoted by a link between two nodes, whereas in (c) each circled group of nodes possess a unique key

the network to conserve energy instead of completing the aggregation protocol. Complete formal proofs of correctness are provided for each protocol.

2 Preliminaries

2.1 Key Pre-Distribution

It is common practice in sensor networks to ensure that two nodes communicate only if they possess or can establish a shared key. To facilitate secure communication, nodes are pre-loaded with cryptographic keys using a *key pre-distribution scheme (KPS)*. Although key pre-distribution schemes are well studied for a variety of sensor network settings, the simplicity of a (N, d)-linear sensor network (e.g., fixed, full-control) lends itself well to three specific key pre-distribution schemes:

1. **Optimal** (N, r)-KPS - Each node shares a pairwise key with nodes at distance 1 and distance r.
2. **Pairwise** (N, r)-KPS - Each node shares a distinct key with each neighbor at distance j, for $1 \leq j \leq r$.
3. **Group-based** (N, r)-KPS - Each consecutive subset of $r + 1$ nodes shares a distinct group key.

The optimal scheme (item 1 in the above list) was introduced by Martin and Paterson [18], and it is optimal with respect to the number of keys necessary to maintain secure connectivity in a network when $d = r$ and up to $r - 1$ nodes are unavailable. If r consecutive nodes are unavailable, then the network becomes disconnected regardless of the underlying KPS. Additionally, each key in the optimal scheme is possessed by exactly two nodes, thus minimizing the number of nodes affected by the compromise of any given key.

The above schemes can be combined, for example, by issuing both pairwise keys and group keys to nodes. Additionally, these basic schemes may be augmented with additional specialized keys, such as each node sharing a unique pairwise key with a base station. In most settings, it is natural to choose $r = d$, although depending on the network setting and requirements, the value of r used for key pre-distrubtion need not match the maximum communication range d of the network.

2.2 Problem Setting

Our goal is to establish resilient data aggregation protocols for linear networks under pairwise, group-based, and optimal key pre-distribution. We assume that aggregation begins at one endpoint of the network and propagates toward a base station at the other endpoint, with each node n_i contributing a single reading r_i from a pre-specified set of possible readings $\mathcal{R} = \{0, 1, \ldots, R - 1\}$. A reading $r_i \in \mathcal{R}$ will be referred to as a *valid* sensor reading, while a reading not in the set $\mathcal{R}$ is *invalid.*

Definition 3 Let $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ be a (N, d)-linear network, let $\mathcal{M} \subseteq \mathcal{N}$ be the set of malicious nodes in $\mathcal{N}$, and let

$$T = \sum_{i=1}^{N} r_i$$

be the actual correct aggregate total of all readings in $\mathcal{N}$. An aggregation protocol is *secure* if it either outputs an aggregate total T' such that

$$|T - T'| \leq |\mathcal{M}|(R - 1),$$

or identifies the presence of one or more malicious nodes.

Definition 3 states that an aggregation protocol is secure if each malicious node can modify the aggregate total by at most a single valid sensor reading. That is, a malicious node can lie about its own reading, but cannot otherwise alter another honest node's reading within the aggregate total. Should the adversary modify the aggregate total by more than a single valid sensor reading, then the base station will be alerted to this fact.

Our model is identical to that of Chan, Perrig, and Song [7], who introduced the notion of a *direct data injection attack*, as an attack where a malicious entity controlling one or more nodes may submit a false reading for each node that it controls, with the constraint that false readings must be valid. This leads to the definition of an *optimally secure* aggregation protocol as a protocol where the base station will not accept any maliciously modified aggregate total, except for what can be achieved using a direct data injection attack.

We assume the presence of an adversary that can eavesdrop on all communication and that can selectively compromise a subset of nodes, but cannot compromise the base station. The adversary learns all key information from compromised nodes and may reprogram or alter the behavior of compromised nodes, but it cannot otherwise alter the physical capabilities of any node. The goal of an adversary controlling k nodes is to modify the aggregate total by more than k valid sensor readings without detection. Attacks such as jamming communication, refusing to participate in the protocol, or other denial-of-service attacks are outside the scope of our threat model. However, such attacks can be detected and addressed through other means, such as placing an upper bound on the running time of the protocol.

3 Secure Aggregation using Optimal KPS

The optimal (N, d)-KPS for (N, d)-linear networks given in [18] specifies that each node shares unique keys with each neighbor at distance 1 and each neighbor at distance d, for a total of four keys per node when $d > 1$, except that nodes less than distance $d - 1$ from an endpoint possess only two keys. This scheme is optimal with respect to the number of keys possessed by each node to maintain secure connectivity even if up to $d - 1$ consecutive nodes fail. In this section, we present a technique for secure aggregation using a KPS inspired by the optimal KPS.

As a starting point, consider a $(N, 1)$-linear network $\mathcal{N} = \{n_1, \ldots, n_N\}$, where each node is capable of communication only with nodes at distance 1. This is the most restrictive case possible, as each node can only speak to its direct neighbors in the network. We assume that some nodes in $\mathcal{N}$ are malicious, but that no more than k consecutive nodes are malicious. Note that we still allow multiple disjoint sets of up to k consecutive malicious nodes, as long as there is at least one honest node in between them. In the same way that the optimal (N, r)-KPS provides resilience against $r - 1$ node failures by ensuring nodes distance r apart have a shared key, we can provide resilience during aggregation against up to k consecutive malicious nodes by ensuring that nodes at distance $k + 1$ have a shared key, even if they are not capable of direct communication. More specifically, although $\mathcal{N}$ is a (N, d)-linear network, we distribute keys using the optimal $(N, k+1)$-KPS. That is, each node shares a distinct key with nodes at distance 1 and nodes at distance $k+1$. The key shared between two nodes n_i and n_j will be referred to by $k_{i,j}$ where $|i - j| = 1$ or $k+1$.

3.1 Aggregation when $k = 1$

To begin, let us consider the case when $k = 1$, i.e., where no two consecutive nodes in $\mathcal{N}$ are both malicious. In order to prevent a single malicious node, say $m_i = n_i$ from contributing an invalid sensor reading, we have node n_{i-1} tag the aggregate total X_{i-1} from the first $i-1$ nodes using a *message authentication code (MAC)* under key $k_{i-1,i+1}$. Node n_{i+1} proceeds if and only if it receives from m_i the value X_{i-1} along with a MAC to verify it is unaltered, as well as the updated aggregate X_i (or, equivalently, the sensor reading r_i). Thus, n_{i+1} can verify that both X_{i-1} and r_i are correct, and, if so, node n_{i+1} can correctly compute $X_{i+1} = X_{i-1} + r_i + r_{i+1}$. Fig. 3 demonstrates this process in three cases. The first case demonstrates the beginning of the protocol, the second case demonstrates non-endpoint nodes, and the third cases demonstrates the termination of the protocol.

The protocol in Fig. 3 is resilient as long as no two malicious nodes are neighbors within the network, because single malicious nodes are forced to pass their reading, along with an authenticated aggregate sum from its previous neighbor, on to an honest node. The honest node verifies the validity of the received reading, adds it into the aggregate sum, and then forwards the sum, its own reading, and an authentication tag on to the next node so the process can be repeated. The protocol terminates either when a node outputs `reject`, or when the base station n_N receives and validates the values X_{N-2} and r_{N-1}. The resilience of the protocol is proven in Theorem 1.

Theorem 1 *Suppose that the protocol in Fig. 3 is being used for data aggregation, that node n_i is honest, that no two neighbors are both dishonest, and that nodes $n_1, \ldots, n_{i-1}$ do not output* `reject`*. Then, n_i either outputs* `reject`*, or node n_i correctly computes $X_i = \sum_{j=1}^{i} r_j$ where $r_1, \ldots, r_i \in \{0, \ldots, R-1\}$.*

Proof We prove this by strong induction on i, first considering $i = 1$ as a base case. If node n_1 is honest, then it knows the correct value of $r_1 = X_1$, and it can compute $t_{1,3} = \mathrm{MAC}_{k_{1,3}}(X_1)$. If node n_1 is dishonest, then node n_2 must be honest. Node n_2 has knowledge of r_2 and receives $X_1 = r_1$ directly from n_1, with which it can verify that $r_1 \in \{0, \ldots, R-1\}$. Therefore, node n_2 can correctly compute $X_2 = r_1 + r_2$ and $t_{2,4} = \mathrm{MAC}_{k_{2,4}}(X_2)$.

For the purpose of strong induction, assume that all honest nodes n_j for $1 \leq j \leq i-1$ can correctly compute

$$X_{j-1} = \sum_{l=1}^{j-1} r_l,$$

where $r_l \in \{0, \ldots, R-1\}$, as well as correctly compute $t_{j-1,j+1}$.

Suppose that node n_{i-1} is honest, and that node n_{i-2} is possibly dishonest. Then, by induction,

$$X_{i-1} = \sum_{l=1}^{i-1} r_l$$

and $r_{i-1} \in \{0, \ldots, R-1\}$. If $t_{i-2,i} \neq \mathrm{MAC}_{k_{i-2,i}}(X_{i-2})$, then node n_i outputs `reject`. Otherwise, $X_{i-2} = X_{i-1} - r_{i-1}$ and node n_i can correctly compute $X_i = X_{i-1} + r_i$ and $t_{i,i+2} = \mathrm{MAC}_{k_{i,i+2}}(X_i)$.

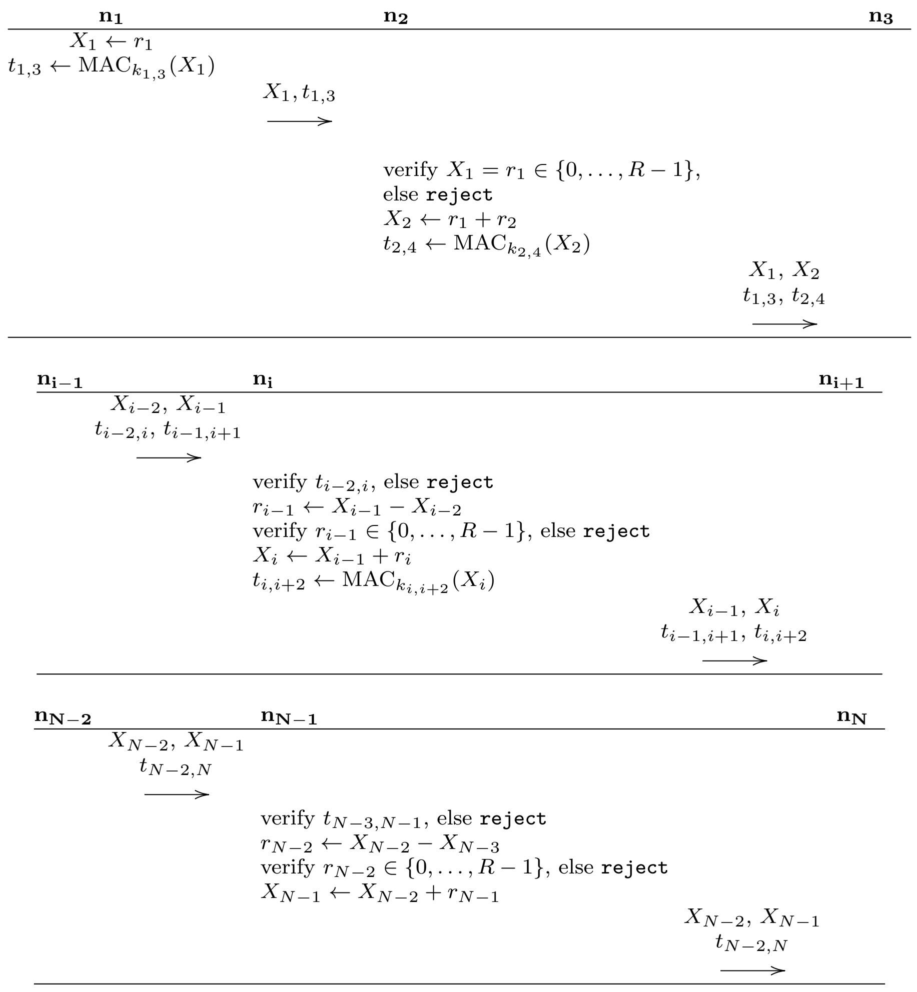
$\mathbf{n_1}$
$\mathbf{n_2}$
$\mathbf{n_3}$
$X_1 \leftarrow r_1$
$t_{1,3} \leftarrow \mathrm{MAC}_{k_{1,3}}(X_1)$
$X_1, t_{1,3}$
verify $X_1 = r_1 \in \{0, \ldots, R-1\}$,
else `reject`
$X_2 \leftarrow r_1 + r_2$
$t_{2,4} \leftarrow \mathrm{MAC}_{k_{2,4}}(X_2)$
$X_1,\ X_2$
$t_{1,3},\ t_{2,4}$
$\mathbf{n_{i-1}}$
$\mathbf{n_i}$
$\mathbf{n_{i+1}}$
$X_{i-2},\ X_{i-1}$
$t_{i-2,i},\ t_{i-1,i+1}$
verify $t_{i-2,i}$, else `reject`
$r_{i-1} \leftarrow X_{i-1} - X_{i-2}$
verify $r_{i-1} \in \{0, \ldots, R-1\}$, else `reject`
$X_i \leftarrow X_{i-1} + r_i$
$t_{i,i+2} \leftarrow \mathrm{MAC}_{k_{i,i+2}}(X_i)$
$X_{i-1},\ X_i$
$t_{i-1,i+1},\ t_{i,i+2}$
$\mathbf{n_{N-2}}$
$\mathbf{n_{N-1}}$
$\mathbf{n_N}$
$X_{N-2},\ X_{N-1}$
$t_{N-2,N}$
verify $t_{N-3,N-1}$, else `reject`
$r_{N-2} \leftarrow X_{N-2} - X_{N-3}$
verify $r_{N-2} \in \{0, \ldots, R-1\}$, else `reject`
$X_{N-1} \leftarrow X_{N-2} + r_{N-1}$
$X_{N-2},\ X_{N-1}$
$t_{N-2,N}$

Fig. 3 A resilient data aggregation protocol for $(N, 1)$-linear networks where $k = 1$ using optimal KPS

Next, suppose that node n_{i-1} is dishonest, then node n_{i-2} must be honest. By induction,

$$X_{i-2} = \sum_{l=1}^{i-2} r_l$$

is the correct aggregate total and $t_{i-2,i}$ is a valid MAC for X_{i-2}. If either of these values are modified by n_{i-1}, then node n_i will output `reject`, unless node n_{i-1} can forge a MAC using key $k_{i-2,i}$. Node n_i receives the value X_{i-1} from node n_{i-1}, computes $r_{i-1} = X_{i-1} - X_{i-2}$, and can verify directly that $r_{i-1} \in \{0, \ldots, R-1\}$. Therefore, node n_i can correctly compute $X_i = X_{i-2} + r_{i-1} + r_i$ and $t_{i,i+2} = \mathrm{MAC}_{k_{i,i+2}}(X_i)$. □

Although the protocol presented in Fig. 3 allows the detection of the presence of active malicious nodes, it does not allow us to precisely identify which node is malicious. If a node n_i fails to verify the tag $t_{k_{i-2,i}}$ there are three possibilities:

1. Node n_i is malicious and falsely claims that $t_{k_{i-2,i}}$ is invalid;
2. Node n_{i-1} is malicious and altered the value $t_{k_{i-2,i}}$; or
3. Node n_{i-2} is malicious and forwarded an invalid $t_{k_{i-2,i}}$.

As these three nodes are the only nodes to handle the value $t_{k_{i-2,i}}$, they are the only nodes that could have possibly altered it. The limited key information and communication range of each node makes detecting which of these cases occurred difficult, if not impossible, without additional assumptions or the assistance of the base station. As our goal is simply to prevent an incorrect aggregate total from being reported, precise adversarial detection and how to respond to it is left as future work.

3.2 Analysis

As a baseline for comparison, we note that any hop-by-hop aggregation protocol requires each node to send at least one message to propagate the aggregate total. Adding an integrity check, such as a MAC, would add an additional message per node. Therefore, we can assume a lower bound of two messages per node and $2N$ messages total in the absence of any malicious nodes.

The communication cost of the $k = 1$ protocol is straightforward to compute. Each non-endpoint node n_i forwards four messages: the current aggregate total X_i and a MAC to be verified by node n_{i+2}, and the previous hop's aggregate total X_{i-1} along with the received MAC to be verified by node n_{i+1}. Therefore, the total communication cost is slightly less than $4N$ messages. Note that in a $(N, 1)$-linear network, communication between nodes at distance 1 is implicitly authenticated when traffic flows in one direction, as there is only one possible source for a received message. A malicious node attempting to impersonate a different honest node would result in duplicate messages, which is readily detected. The protocol could be altered to explicitly authenticate messages between nodes at distance 1, either by adding an additional MAC (yielding a total of five messages per node), or by utilizing an authenticated mode of encryption. The latter case is preferable, as it does not require an additional message to be sent.

For ease of presentation, no additional information was included in any MAC to avoid *replay attacks*. If the protocol is run more than once, then an adversary can reuse messages and MACs from a prior run of the protocol without detection. In practice, a *nonce* must be included in each MAC to ensure freshness. Depending on the specific application, a nonce is readily available in the form a counter, timestamp, or session identifier.

3.3 An Attack Against a Naive $k > 1$ Protocol

The protocol in the previous section can be naturally extended to larger values of k simply by distributing keys using the optimal $(N, k + 1)$-KPS. Each node would then receive an authenticated aggregate sum from $k + 1$ hops back, along with readings from the previous k hops to verify directly.

Fig. 4 demonstrates such an approach for $k = 2$. Unfortunately, this method is not adequate to protect against colluding malicious nodes.

Although this approach is the natural generalization of the secure $k = 1$ protocol presented in the previous section, it is now possible for a pair of malicious nodes to alter the aggregate total by more than two valid sensor readings. This attack arises due to the fact that, in the $k > 1$ setting, an honest node immediately preceding a malicious node is not guaranteed to have its aggregate total verified by an honest node. For example, if node n_i is honest, it is possible that nodes n_{i+1} and n_{i+3} are both dishonest. If this occurs, it is possible for node n_{i+1} to alter the total without being detected, as malicious node n_{i+3} would normally be the node to detect such an attack. This situation could not arise in the $k = 1$ case, as any malicious node always has an honest node on either side of it. In other words, this approach works to defend against two consecutive malicious nodes, but no longer protects against colluding non-consecutive malicious nodes.

To illustrate an attack against the naive $k = 2$ protocol, consider a network where the honest reading of node n_i is $r_i = 0$. A pair of malicious nodes should be able to modify the total by at most $2(R - 1)$. Consider a connected subset of the network $\{n_i, m_{i+1},$

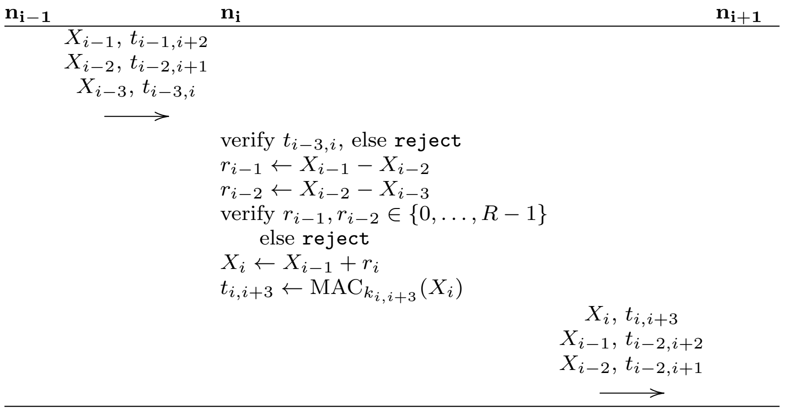
$\mathbf{n_{i-1}}$
$\mathbf{n_i}$
$\mathbf{n_{i+1}}$
$X_{i-1},\ t_{i-1,i+2}$
$X_{i-2},\ t_{i-2,i+1}$
$X_{i-3},\ t_{i-3,i}$
verify $t_{i-3,i}$, else reject
$r_{i-1} \leftarrow X_{i-1} - X_{i-2}$
$r_{i-2} \leftarrow X_{i-2} - X_{i-3}$
verify $r_{i-1}, r_{i-2} \in \{0, \ldots, R-1\}$
else reject
$X_i \leftarrow X_{i-1} + r_i$
$t_{i,i+3} \leftarrow \mathrm{MAC}_{k_{i,i+3}}(X_i)$
$X_i,\ t_{i,i+3}$
$X_{i-1},\ t_{i-2,i+2}$
$X_{i-2},\ t_{i-2,i+1}$

Fig. 4 A flawed data aggregation protocol for $(N, 1)$-linear networks where $k = 2$

n_{i+2}, m_{i+3}, $n_{i+4}\}$, where n_k denotes an honest node and m_k denotes a malicious node. The attack proceeds as follows:

1. Node n_i sends the following to node m_{i+1}:
$$X_{i-2} = X_{i-1} = X_i = 0$$
$$t_{i-2,i+1}, t_{i-1,i+2}, t_{i,i+3}.$$
2. Node m_{i+1} replaces X_i with $X'_i = R - 1$, adds the reading $r_{i+1} = R - 1$ to the total such that $X_{i+1} = 2(R-1)$, and computes the MAC $t_{i+1,i+4}$ on this total. The following is forwarded to node n_{i+2}:
$$X_{i-1} = 0, X'_i = R - 1, X_{i+1} = 2(R - 1)$$
$$t_{i-1,i+2}, t_{i,i+3}, t_{i+1,i+4}.$$
3. Node n_{i+2} verifies the unaltered value $X_{i-1} = 0$ and accepts, forwarding the following to node m_{i+3}:
$$X'_i = R - 1, X_{i+1} = 2(R - 1), X_{i+2} = 2(R - 1)$$
$$t_{i,i+3}, t_{i+1,i+4}, t_{i+2,i+5}$$
4. Node m_{i+3} is supposed to verify that $t_{i,i+3}$ is a valid MAC (on $X_i = 0$), but ignores the fact that the malicious node m_{i+1} has replaced it with $X'_i = R - 1$. Node m_{i+3} sets $r_{i+3} = R - 1$, computes $X_{i+3} = 3(R - 1)$, and computes a MAC $t_{i+3,i+6}$ on this value. The following is forwarded to node n_{i+4}:
$$X_{i+1} = X_{i+2} = 2(R - 1), X_{i+3} = 3(R - 1)$$
$$t_{i+1,i+4}, t_{i+2,i+5}, t_{i+3,i+6}$$
5. Node n_{i+4} verifies that $t_{i+1,i+4}$ is a valid MAC on X_{i+1} and accepts X_{i+1} as valid. At this point, the malicious nodes have managed to modify the total by $3(R-1)$ without being detected.

This attack can be avoided if any MAC generated by an honest node is always verified by an honest node, however this requires a much stronger assumption on the distribution of malicious nodes when working in the optimal KPS setting, that is unlikely to apply in any practical setting. The next section demonstrates how this problem can be overcome if additional keys are distributed to nodes, such as in the pairwise KPS model.

4 Aggregation Using Pairwise KPS

The previous section presented a protocol for detecting non-consecutive malicious nodes during aggregation and demonstrated the difficulty in extending it to detecting coalitions of malicious nodes. In this section we extend the protocol to protect against a coalition of up to k consecutive malicious nodes when $k > 1$. The goal of this protocol remains the same: to prevent a malicious node from altering the aggregate total by more than a single valid sensor reading. A coalition of consecutive malicious nodes should therefore only be able to modify the aggregate total by no more than k valid sensor readings.

The basic idea behind the single-node protocol is to exploit the existence of honest nodes on either side of any malicious node. The first honest node computes and forwards an authenticated aggregate total that the next honest node can verify. If the malicious node in the middle misbehaves, one of the next two hops will detect it. The k-resilient version of the protocol is based on the same idea using the pairwise $(N, k+1)$-KPS. Any coalition of up to k consecutive nodes must have an honest node on either side of it. Thus, authentication information from $k+1, k, \ldots, 2$ hops back will be forwarded at each hop to ensure that none of the preceding k nodes have altered the aggregate total in an invalid manner. Therefore, the extended protocol requires that any two nodes n_i and n_j within distance $d \leq k+1$ of each other must have a unique pairwise key, $k_{i,j}$, shared with each other. Fig. 5 demonstrates a 2-resilient aggregation protocol, broken into three cases demonstrating the beginning, general case, and termination of the protocol. A security proof when $k = 2$ follows.

The protocol presented here assumes a $(N, 1)$-linear network with a pairwise $(N, k+1)$-KPS. In an $(N, k+1)$-linear network, the protocol can be made much more efficient by node i forwarding the value $t_{i,i+k+1}$ directly to node n_{i+k+1}. In general, for any communication range r where $1 < r < k+1$, each node will forward messages directly to any node at distance $d \leq r$, or r hops further in the network for nodes at distance $d > r$. As in the $k = 1$ protocol, if the communication range of nodes is only 1 (i.e., a $(N, 1)$-linear network), then a separate MAC may not be necessary to authenticate messages between nodes at distance $d = 1$.

Theorem 2 *Suppose the protocol in Fig. 5 is being used for data aggregation, that node n_i is honest, that at most $k = 2$ consecutive nodes are dishonest, and that nodes $n_1, \ldots, n_{i-1}$ do not output `reject`. Then, n_i either outputs `reject`, or node n_i correctly computes $X_i = \sum_{j=1}^{i} r_j$ where $r_1, \ldots, r_i \in \{0, \ldots, R-1\}$.*

Proof We prove this by strong induction on i, first considering $i = 1, 2, 3$ as base cases.

- If node n_1 is honest, then it knows the correct value of $r_1 = X_1$, and it can correctly compute

$$\begin{aligned} t_{1,4} &= \mathrm{MAC}_{k_{1,4}}(X_1), \\ t_{1,3} &= \mathrm{MAC}_{k_{1,3}}(X_1), \text{ and} \\ t_{1,2} &= \mathrm{MAC}_{k_{1,2}}(X_1). \end{aligned}$$

- If node n_1 is dishonest, then node n_2 may be honest. If so, node n_2 receives $X_1 = r_1$ directly from n_1, with which it can verify $t_{1,2}$ is valid and that $r_1 \in \{0, \ldots, R-1\}$, outputting `reject` if not. Therefore, node n_2 can correctly compute

$$\begin{aligned} X_2 &= r_1 + r_2, \\ t_{2,5} &= \mathrm{MAC}_{k_{2,5}}(X_2), \\ t_{2,4} &= \mathrm{MAC}_{k_{2,4}}(X_2), \text{ and} \\ t_{2,3} &= \mathrm{MAC}_{k_{2,3}}(X_2), \end{aligned}$$

  or outputs `reject`.

$\mathbf{n_1}$		$\mathbf{n_2}$		$\mathbf{n_3}$
$X_1 \leftarrow r_1$ $t_{1,2} \leftarrow \mathrm{MAC}_{k_{1,2}}(X_i)$ $t_{1,3} \leftarrow \mathrm{MAC}_{k_{1,3}}(X_i)$ $t_{1,4} \leftarrow \mathrm{MAC}_{k_{1,4}}(X_i)$				
	$X_1,\, t_{1,2},\, t_{1,3},\, t_{1,4}$ $\xrightarrow{\hspace{2em}}$			
		verify $t_{1,2}$, else `reject` verify $r_{i-1} = X_1 \in \{0, \ldots, R-1\}$, else `reject` $X_2 \leftarrow X_1 + r_2$ $t_{2,3} \leftarrow \mathrm{MAC}_{k_{2,3}}(X_2)$ $t_{2,4} \leftarrow \mathrm{MAC}_{k_{2,4}}(X_2)$ $t_{2,5} \leftarrow \mathrm{MAC}_{k_{2,5}}(X_2)$		
			$X_1,\, t_{1,3},\, t_{1,4}$ $X_2,\, t_{2,3},\, t_{2,4},\, X_{2,5}$ $\xrightarrow{\hspace{2em}}$	

$\mathbf{n_{i-1}}$		$\mathbf{n_i}$		$\mathbf{n_{i+1}}$
	$X_{i-3},\, t_{i-3,i}$ $X_{i-2},\, t_{i-2,i},\, t_{i-2,i+1}$ $X_{i-1},\, t_{i-1,i},\, t_{i-1,i+1},\, t_{i-1,i+2}$ $\xrightarrow{\hspace{2em}}$			
		verify $t_{i-1,i},\, t_{i-2,i},\, t_{i-3,i}$ else `reject` $r_{i-1} \leftarrow X_{i-1} - X_{i-2}$ $r_{i-2} \leftarrow X_{i-2} - X_{i-3}$ verify $r_{i-1} \in \{0, \ldots, R-1\}$, else `reject` verify $r_{i-2} \in \{0, \ldots, R-1\}$, else `reject` $X_i \leftarrow X_{i-1} + r_i$ $t_{i,i+1} \leftarrow \mathrm{MAC}_{k_{i,i+1}}(X_i)$ $t_{i,i+2} \leftarrow \mathrm{MAC}_{k_{i,i+2}}(X_i)$ $t_{i,i+3} \leftarrow \mathrm{MAC}_{k_{i,i+3}}(X_i)$		
			$X_{i-2},\, t_{i-2,i+1}$ $X_{i-1},\, t_{i-1,i+1},\, t_{i-1,i+2}$ $X_i,\, t_{i,i+1},\, t_{i,i+2},\, t_{i,i+3}$ $\xrightarrow{\hspace{2em}}$	

$\mathbf{n_{N-2}}$		$\mathbf{n_{N-1}}$		$\mathbf{n_N}$
	$X_{N-4},\, t_{N-4,N-1}$ $X_{N-3},\, t_{N-3,N-1},\, t_{N-3,N}$ $X_{i-1},\, t_{N-2,N-1},\, t_{N-2,N},$ $\xrightarrow{\hspace{2em}}$			
		verify $t_{N-2,N-1},\, t_{N-3,N-1},\, t_{N-4,N-1}$ else `reject` $r_{N-2} \leftarrow X_{N-2} - X_{N-3}$ $r_{N-3} \leftarrow X_{N-3} - X_{N-4}$ verify $r_{N-2} \in \{0, \ldots, R-1\}$, else `reject` verify $r_{N-3} \in \{0, \ldots, R-1\}$, else `reject` $X_{N-1} \leftarrow X_{N-2} + r_{N-1}$ $t_{N-1,N} \leftarrow \mathrm{MAC}_{k_{N-1,N}}(X_i)$		
			$X_{N-3},\, t_{N-3,N}$ $X_{N-2},\, t_{N-2,N}$ $X_{N-1},\, t_{N-1,N}$ $\xrightarrow{\hspace{2em}}$	

Fig. 5 A resilient data aggregation protocol for $k = 2$. Each node receives the aggregate sum from the three nodes preceding it, along with a MAC from neighbors at distance $d = 1, 2, 3$ to ensure that the sums are unaltered by any potentially malicious intermediate nodes

– If both nodes n_1 and n_2 are dishonest, then node n_3 must be honest. Node n_3 receives $X_1, t_{1,3}, t_{1,4}, X_2, t_{2,3}, t_{2,4}, t_{2,5}$. From these values, node n_3 can verify that $t_{1,3}$ and $t_{2,3}$ are valid and compute $r_1 = X_1$ and $r_2 = X_2 - X_1$, verifying that $r_1, r_2 \in \{0, \ldots, R-1\}$, and outputting `reject` if not. Therefore, node n_3 can correctly compute

$$\begin{aligned} X_3 &= r_1 + r_2 + r_3, \\ t_{3,6} &= \mathrm{MAC}_{k_{3,6}}(X_3), \\ t_{3,5} &= \mathrm{MAC}_{k_{3,5}}(X_3), \text{ and} \\ t_{3,4} &= \mathrm{MAC}_{k_{3,4}}(X_3), \end{aligned}$$

or outputs `reject`.

For the purpose of strong induction, assume an honest node n_j for $1 \leq j \leq i-1$ can correctly compute

$$X_{j-1} = \sum_{l=1}^{j-1} r_l,$$

where $r_l \in \{0, \ldots, R-1\}$, as well as correctly compute $t_{i-1,i+2}$, $t_{i-1,i+1}$, $t_{i-1,i}$.

Suppose that node n_{i-1} is honest, and that nodes n_{i-2} and n_{i-3} are possibly dishonest. Then, by induction,

$$X_{i-1} = \sum_{l=1}^{i-1} r_l,$$

where $r_l \in \{0, \ldots, R-1\}$. If $t_{i-3,i}$, $t_{i-2,i}$, and $t_{i-1,i}$ are not valid MACs, then node n_i outputs `reject`. Otherwise, the received values X_{i-3}, X_{i-2}, and X_{i-1} are the same values forwarded by nodes n_{i-3}, n_{i-2}, and n_{i-1} respectively. Because node n_{i-1} is honest, $r_{i-1} = X_{i-1} - X_{i-2} \in \{0, \ldots, R-1\}$, and node n_i can correctly compute

$$\begin{aligned} X_i &= X_{i-1} + r_i, \\ t_{i,i+1} &= \mathrm{MAC}_{k_{i,i+1}}(X_i), \\ t_{i,i+2} &= \mathrm{MAC}_{k_{i,i+2}}(X_i), \text{ and} \\ t_{i,i+3} &= \mathrm{MAC}_{k_{i,i+3}}(X_i). \end{aligned}$$

Next, suppose that node n_{i-1} is dishonest and that node n_{i-2} is honest. Then, by induction,

$$X_{i-2} = \sum_{l=1}^{i-2} r_l$$

is the correct aggregate sum at node n_{i-2}, and $t_{i-2,i}$ is a valid MAC for X_{i-2}. If $t_{i-3,i}$, $t_{i-2,i}$, or $t_{i-1,i}$ are invalid, then node n_i outputs `reject`. Otherwise, the received values X_{i-3} and X_{i-2} are the same values forwarded by nodes n_{i-3} and n_{i-2} respectively. Therefore, node n_i can correctly compute $r_{i-1} = X_{i-1} - X_{i-2}$ and verify that $r_{i-1} \in \{0, \ldots, R-1\}$, outputting `reject` if not. If node n_i does not output `reject`, then X_{i-1} is correct and node n_i can compute

$$\begin{aligned} X_i &= X_{i-1} + r_i, \\ t_{i,i+1} &= \mathrm{MAC}_{k_{i,i+1}}(X_i), \\ t_{i,i+2} &= \mathrm{MAC}_{k_{i,i+2}}(X_i), \text{ and} \\ t_{i,i+3} &= \mathrm{MAC}_{k_{i,i+3}}(X_i). \end{aligned}$$

Finally, suppose that both nodes n_{i-1} and n_{i-2} are dishonest. Then, by assumption, node n_{i-3} must be honest, and, by induction,

$$X_{i-3} = \sum_{l=1}^{i-3} r_l$$

is the correct aggregate sum at node n_{i-3}, and $t_{i-3,i}$ is a valid MAC for X_{i-3}. If $t_{i-3,i}$, $t_{i-2,i}$, or $t_{i-1,i}$ are invalid, then node n_i outputs `reject`. Otherwise, node n_i can compute

$r_{i-2} = X_{i-2} - X_{i-3}$ and $r_{i-1} = X_{i-1} - X_{i-2}$ and verify $r_{i-2}, r_{i-1} \in \{0, \ldots, R-1\}$, outputting `reject` if not. If node n_i does not output `reject`, then X_{i-1} and X_{i-2} are correct and node n_i can compute

$$\begin{aligned} X_i &= X_{i-1} + r_i, \\ t_{i,i+1} &= \mathrm{MAC}_{k_{i,i+1}}(X_i), \\ t_{i,i+2} &= \mathrm{MAC}_{k_{i,i+2}}(X_i), \text{ and} \\ t_{i,i+3} &= \mathrm{MAC}_{k_{i,i+3}}(X_i). \end{aligned}$$

□

This protocol generalizes naturally to larger values of k. Each node receives the aggregate total for each of the preceding $k+1$ nodes, along with a MAC on each of them to very authenticity. From this information, a node can compute the previous k sensor readings, verify that they are valid, and compute the updated aggregate total. The node then computes a MAC on the updated aggregate total for each of the $k+1$ nodes at distance $1, 2, 3, \ldots, k+1$. The security proof is similar to the $k = 2$ case.

4.1 Analysis

The communication cost of each node is dependent on both the maximum number of consecutive malicious nodes k, and the maximum communication range of each node d.

To begin, assume that $d = 1$, and each message must be routed hop-by-hop. Each node n_i must forward the current aggregate total, as well as authentication information for $k+1$ nodes at distance $1, 2, \ldots k+1$ nodes further in the network. Therefore, node n_i creates and sends X_i as well as k MACs on X_i to the next hop (giving a total of $k+2$ messages).

In addition to the messages node n_i generates, it is also responsible for forwarding authentication information from the previous k hops:

- The aggregate total X_{i-k} along with one MAC to be forwarded for node n_{i+1} (two messages).
- The aggregate total X_{i-k+1} along with two MACs to be forwarded for nodes n_{i+1} and n_{i+1} (three messages).
- $\vdots$
- The aggregate total X_{i-1} along with k MACs to be verified by nodes n_{i+1} through n_{i+k} ($k+1$ messages).

Therefore, in addition to the $k+2$ messages node n_i generates, it also forwards

$$\begin{aligned} 2 + 3 + \ldots + k + 1 &= \frac{(k+1)(k+2)}{2} - 1 \\ &= \frac{(k^2 + 3k + 2)}{2} - 1 \\ &= \frac{1}{2}(k^2 + 3k) \end{aligned}$$

messages from previous hops. Therefore, the total communication cost when $d = 1$ for node n_i is

$$\frac{1}{2}(k^2 + 3k) + k + 2 = \frac{1}{2}(k^2 + 5k + 4)$$

messages.

Next, assume that $1 < d \leq k+1$. In this case, each message is forwarded to the correct node if it is within communication range, or d hops further in the network towards its destination. In order to approximate the total communication cost, we consider the

Max cons. malicious nodes	Communication cost (per node)
$k = 1$	$\frac{1}{2}(k^2 + 5k + 4)$
$k < d$ and $k \mid d$	$\frac{k^2+3kd+2k+3d+1}{2d}$
$k < d$ and $k \nmid d$	$\frac{k^2+3kd+2k+3d+1}{2d} + (\lceil \frac{k+1}{d} \rceil)(k+1 \mod d)$
$k = d - 1$	$2k + 2$

Fig. 6 Per node communication cost for resilient aggregation using the pairwise KPS

amount of network traffic a single node not near an endpoint generates when forwarding authentication information to the next $k + 1$ nodes in the network.

For ease of presentation, assume $k + 1$ is a multiple of d. Then messages may need to travel up to $h = \frac{k+1}{d}$ hops in the network during aggregation.

- Authentication information sent to nodes up to distance d away is sent directly (d messages in total).
- Authentication information sent to nodes between distance $d + 1$ and $2d$ travels *two* hops ($2d$ messages in total).
- $\vdots$
- Authentication information sent to nodes between distance $(h - 1)d$ and hd travels h hops (hd messages in total).

Therefore, the authentication information generated by a node incurs a communication cost of

$$\begin{aligned} d + 2d + \ldots + hd &= \frac{h(h+1)}{2}d \\ &= \frac{\frac{k+1}{d}\left(\frac{k+1}{d}+1\right)}{2}d \\ &= \frac{(k+1)(k+d+1)}{2d} \\ &= \frac{k^2+kd+2k+d+1}{2d} \end{aligned}$$

If $k + 1$ is not a multiple of d, then there are an additional $k + 1 \mod d$ authentication messages that must travel $\lceil \frac{k+1}{d} \rceil$ hops.

Each node also forwards the aggregate total for itself to the next $k+1$ hops, generating an additional $k + 1$ messages. Therefore, the total communication cost incurred by each node is

$$\frac{k^2+3kd+2k+3d+1}{2d} + (\lceil \frac{k+1}{d} \rceil)(k+1 \mod d).$$

Because each node sends and receives the same number of messages (except for those within $k + 1$ hops of an endpoint), the amount of traffic a non-endpoint node generates is identical to the number of messages it is responsible for forwarding during a run of the protocol. Therefore, the expression above describes the total number of messages each non-endpoint node must send, as well as providing an upper-bound on the number of messages nodes near an end-point must send. The total communication cost of aggregation using the pairwise KPS is summarized in Fig. 6.

As in the previous aggregation protocol, an application-specific nonce must be included in each MAC to prevent replay attacks.

5 Aggregation Using Group-Based KPS

In group-based key pre-distribution, each node is a member of one or more *groups*, and each group has an associated *group key* to allow secure communication among group members.

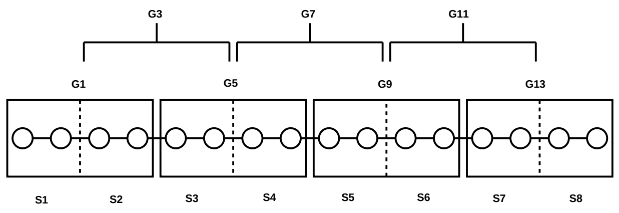

Fig. 7 A $(16, 3)$-linear network partitioned into groups and subgroups

In an (N, d)-linear network, it is natural to divide the network into connected groups of size $d + 1$, as this is the largest group size such that all members of a group are within each other's communication range. We refer to the group containing nodes $n_i, \ldots, n_{i+d}$ as group g_i, the members of which all possess the shared group key k_i.

Let $\mathcal{N}$ be a (N, d)-linear network where $d > 1$ and group keys are distributed as described above. For ease of presentation, assume d is odd and N is a multiple of $d + 1$. This allows $\mathcal{N}$ to be uniquely partitioned into disjoint connected *subgroups* of size $\frac{d+1}{2}$:

$$\mathcal{N} = s_1 \cup s_2 \cup \ldots \cup s_{2\frac{N}{d+1}},$$

where

$$s_i = \left\{ n_{\frac{(i-1)(d+1)}{2}+1}, n_{\frac{(i-1)(d+1)}{2}+2}, \ldots, n_{\frac{i(d+1)}{2}} \right\}$$

and $s_i \cup s_{i+1}$ is a group in the underlying group-based KPS possessing some group key

$$sk_{i,i+1} = k_{\frac{(i-1)(d+1)}{2}+1}.$$

For example, consider a $(16, 3)$-linear network partitioned under this scheme. The 16 nodes are partitioned into four groups of adjacent nodes:

$$\begin{array}{ll} g_1 = \{n_1, n_2, n_3, n_4\} & g_5 = \{n_5, n_6, n_7, n_8\} \\ g_9 = \{n_9, n_{10}, n_{11}, n_{12}\} & g_{13} = \{n_{13}, n_{14}, n_{15}, n_{16}\}. \end{array}$$

Each subroup is further partioned into two subgroups:

$$\begin{array}{lll} s_1 = \{n_1, n_2\} & s_2 = \{n_3, n_4\} & s_3 = \{n_5, n_6\} \\ s_4 = \{n_7, n_8\} & s_5 = \{n_9, n_{10}\} & s_6 = \{n_{11}, n_{12}\} \\ s_7 = \{n_{13}, n_{14}\} & s_8 = \{n_{15}, n_{16}\}. & \end{array}$$

In this scheme, $s_1 \cup s_2 = g_1$ in the underlying group-based KPS, so all members of $s_1 \cup s_2$ share the key $sk_{1,2} = k_1$. Similarly, $s_2 \cup s_3 = g_3$ in the underlying KPS, with members of both subgroups possessing the key $sk_{2,3} = k_3$.

Because all members of a group (or subgroup) share a common key, aggregation within a group (or subgroup) is trivial. Each member can simply broadcast its reading using the group key, and all members of the group can independently verify the validity of each sensor reading, as well as compute the aggregate total for the group. If any sensor reading is out of range, then the result of the protocol is `reject`. This is accomplished using only a single message from each node within the group. We refer to this sub-protocol as *in-group aggregation.*

Next, to perform aggregation across the entire network, each subgroup of nodes s_i first computes its in-group aggregate total Y_{s_i}. Aggregation then occurs subgroup by subgroup along the network, with each node from a subgroup updating the aggregate total with the current subgroup aggregate, and then forwarding the result to the next subgroup. When passing the aggregate total X_i from subgroup s_i to subgroup s_{i+1}, the key $sk_{i,i+1}$ is used, ensuring that all members of s_{i+1} receive the aggregate total. This also allows all other members of s_i to independently verify that the correct aggregate value was

```
Each $n_j \in s_i$ performs the following:
1. broadcast $(n_j, r_j)$ encrypted under key $sk_{i,i+1}$
2. set $Y_{s_i} \leftarrow r_j$
3. for each $n_l \in s_i - \{n_j\}$ do
   (a) receive $(n_l, r_l)$ encrypted under key $sk_{i,i+1}$
   (b) if $r_l \in \mathcal{R}$ then $Y_{s_i} \leftarrow Y_{s_i} + r_l$
4. resolve duplicate readings / bad readings
5. output $Y_{s_i}$ (the aggregate total of all non-rejected readings)
```

Fig. 8 In-group aggregation for subgroup s_i. Upon completion, each node in s_i correctly learns the subgroup aggregate Y_{s_i} or outputs `reject` for one or more nodes

```
for each $s_i \in \mathcal{N}$ do
    for each $n_j \in s_i$
    1. receive $(n_l, X_{i-1})$ for $n_l \in s_{i-1}$
    2. $X_{i-1} \leftarrow \texttt{SELECT}\{(n_l, X_{i-1}) | n_l \in s_{i-1}\}$
    3. $Y_{s_i} \leftarrow$ In-Group-Agg($s_i$)
    4. $X_i \leftarrow X_{i-1} + Y_{s_i}$
    5. Broadcast $(n_j, X_i)$ encrypted under key $sk_{i,i+1}$
```

Fig. 9 Group-based aggregation protocol

passed on by each node. As long as a majority of nodes within each subgroup are honest, a malicious coalition of nodes will not be able to forward an incorrect aggregate total without detection. Fig. 9 demonstrates this process. In the case that malicious behavior is detected, the honest nodes could choose to continue and re-run the protocol, with those nodes that did not agree with the majority omitted during the second run. An alternative approach is to halt aggregation and forward the labels of the detected malicious nodes to the base station.

The protocol in Fig. 9 is broken into several steps. During the first step, nodes in subgroup s_i are informed of the aggregate total X_{i-1} by each node in s_{i-1}. In the second step, each node applies a `SELECT` function to determine the correct value of X_{i-1} in the case that all readings are not identical. A possible `SELECT` function is discussed below. Next, nodes in subgroup s_i perform in-group aggregation to determine Y_{s_i}. Finally, each node computes the updated aggregate total X_i and broadcasts the result, thereby allowing the next subgroup to continue with the protocol.

In step 1 it is possible that a malicious node m_l may choose to forward an incorrect aggregate total (m_l, X'_{i-1}). The job of the `SELECT` function is to take all received (n_l, X_{i-1}) tuples and output the correct value of X_{i-1}. A simple and natural choice for the `SELECT` function is to choose the aggregate total that a majority of nodes agree on, or output `REJECT` if no majority exists. With such a `SELECT` function, the correct aggregate total can be computed with complexity $O(d)$, and it can be shown that the group-based aggregation protocol output the correct aggregate total whenever a majority of nodes in each subgroup is not malicious. This is demonstrated in Theorem 4.

Both the in-group aggregation and group-based aggregation protocols are vulnerable to message flooding or node spoofing attacks, where a malicious node submits more than one reading for itself or while pretending to be another node. Without additional shared keys or assumptions, defending against such an attack in this setting is impossible. Techniques for mitigating these attacks are discussed in Sect. 5.1. The proofs below demonstrate security in the absence of spoofing attacks.

Theorem 3 *If nodes cannot spoof their identity, then the in-group aggregation protocol in Fig. 8 allows each node in a subgroup s_i to correctly compute the subgroup aggregate total Y_{s_i}, such that Y_{s_i} contains a valid reading from each honest node in s_i, and at most one* *valid sensor reading for each dishonest node in s_i. If any invalid readings are received, the protocol also outputs* `reject`.

Proof Suppose each node in $n_j \in s_i$ broadcasts a valid sensor reading r_j using the key $k_{i,i+1}$. Therefore, each node in s_i receives r_j directly, verifies $r_j \in \mathcal{R}$, and adds r_j to Y_{s_i}. On the other hand, if a dishonest n_j broadcasts $r_j \notin \mathcal{R}$, then it is omitted from the aggregate total and the node n_j is marked as malicious. Thus, only valid readings from dishonest nodes are included in Y_{s_i}. $\square$

Theorem 4 *Suppose a majority of nodes in each subgroup are honest, nodes cannot spoof their identity, and subgroups $s_1, \ldots, s_{i-1}$ do not output* `reject`*. Then, using the protocol in Fig. 9 with the* `SELECT` *function choosing the majority answer, each node in s_i learns the correct aggregate total $X_i = \sum_{j=1}^{i} Y_{s_j}$.*

Proof Nodes in s_1 can run the in-group aggregation protocol, and, by Theorem 3, they correctly learn the aggregate total $Y_{s_1} = X_1$.

For the purpose of induction, assume nodes in s_{i-1} can correctly compute $Y_{s_{i-1}}$ and X_{i-1}, where $Y_{s_{i-1}}$ is the subgroup aggregate for s_{i-1} and X_{i-1} is the aggregate total $\sum_{j=1}^{i-1} X_{s_j}$.

If a node $n_j \in s_{i-1}$ is honest, then it broadcasts the correct message X_{i-1} using key $k_{i,i+1}$, which is received by each node in s_i. Because a majority of nodes in s_{i-1} are honest, the output of `SELECT`$\{(n_j, X_{i-1}) | n_j \in s_{i-1}\}$ is the correct aggregate total X_{i-1}. Therefore, each node in s_i will accept the correct value X_{i-1}. By Theorem 3, each node in s_i can correctly compute Y_{s_i}. Therefore, each node can correctly compute $X_i = X_{i-1} + Y_{s_i}$. $\square$

5.1 Reacting to Node Spoofing Attacks

The in-group aggregation and group-based aggregation protocols in this section rely on the assumption that each node can contribute only a single reading or aggregate total in each protocol. In particular, the protocols are not secure if a malicious node m_i can pretend to be an honest node n_j. In this case, the malicious node can spoof n_j's identity by broadcasting a message of the form (n_j, r_j'), causing all nodes to receive readings of both r_j and r_j' for node n_j. Therefore, a mechanism to react to duplicate readings must be in place.

The node spoofing attack is possible due to the lack of source authentication when sending messages using group keys. There are several approaches that can be used to mitigate this, depending on the individual capabilities of each node. Some potential solutions are discussed below.

5.1.1 Pairwise Keys

Assume that each pair of nodes in any group possess a unique shared key. If two or more readings are received for a given node n_i, then then node n_i can prove which reading is correct by sending a MAC on (n_i, r_i) to each member of the subgroup individually using its pairwise keys instead of the group key. Thus, each node learns the correct value r_i and can continue with the protocol. If multiple valid MACs are received, then it can be assumed that node n_i is compromised or malicious.

5.1.2 Wireless Fingerprinting

Wireless fingerprinting [22] allows one node to generate a deterministic identifier from the physical characteristics of any received message from another node. If fingerprinting is available, then two identical messages from different senders are distinguishable. There are two approaches to using fingerprinting: pre-shared fingerprints, and discovered fingerprints.

In the former case, each node is pre-loaded with the fingerprint of each other node in the group, while in the latter, each node learns the fingerprint of a node after deployment.

In the case of pre-shared fingerprints, determining the true reading from a set of duplicates is trivial. If multiple messages match a single fingerprint, it can be assumed that the node is malicious. When fingerprints are not pre-loaded, during the first run of the protocol each node can record the fingerprint associated with the received message for each node. Because fingerprints are unique, each node can commit to at most one identity within the network. If one fingerprint is associated with multiple nodes, then it can be assumed that the fingerprint belongs to a malicious node. If two different fingerprints commit to the same identity there may be no possible way to tell which one is honest. In this case, both nodes can be marked as malicious as a precaution. This limits the impact of a malicious node to knocking out a single honest node.

5.1.3 Directional Antennae and Distance Bounding

Directional antennae [2] allow a node to estimate the direction of the sender of a message, which is easier to accomplish in a linear network than in a two-dimensional network. If duplicate messages are received, any message that does not originate from the correct side of the network (i.e., left or right) cannot possibly be from an honest node.

Similarly, distance bounding [23] allows a node to put an upper bound on the distance of a sender. If nodes are homogeneous, uniformly spaced, and broadcast at fixed power levels, precise distance location, rather than just an upper bound, may also be available. If duplicate messages are received, any message that does not originate from a node within the correct distance cannot possibly be from an honest node.

On their own, neither direction nor distance bound information is sufficient to determine which message from a set of duplicates came from the correct sender. However, direction and precise distance trivially solve the problem, as only a single node can be at a given distance and direction. Precise distance on its own is sufficient as well, however, determining the correct message requires all nodes in a group to participate in the process. This is possible due to the fact that all nodes in the next subgroup are always located later in the network, and, by assumption, this subgroup contains an honest majority of nodes. If nodes in the next subgroup also confirm the measured distances of a given message, there will be an honest majority of nodes agreeing on a specific location.

6 Concluding Remarks

In a non-adversarial setting, resilient hop-by-hop aggregation in a simple linear network can be performed using N messages to aggregate each reading hop-by-hop, and an additional N messages to include a single homomorphic MAC to detect any errors. Therefore, we can use a baseline of $2N$ messages as a point of comparison for our protocols. Our results are summarized in Fig. 10.

Using the optimal KPS, we presented a protocol that protects against any number of malicious nodes, provided that no two malicious nodes are side by side in the network. This is achieved with only four keys per node, and requires each node to send two messages (a single reading, and the aggregate reading), along with two authentication tags. The total cost of this protocol is $4N$ messages, with two MACs being computed by each node. The limited number of keys in this setting makes extending such an approach to a larger number of adjacent malicious nodes difficult.

Using the pairwise KPS, we presented a protocol that protects against k adjacent malicious nodes within the network. The cost of this protocol scales with the value of k. As k grows, information from $k + 1$ hops back, along with the associated authentication information, must be passed along the network and verified by each node. Depending on the communication range of the nodes, the total communication complexity of the

Malicious nodes	Communication cost (per node)
Optimal KPS	
$k = 1$	4
Pairwise KPS	
$k = 1$	$\frac{1}{2}(k^2 + 5k + 4)$
$k < d$ and $k \mid d$	$\frac{k^2+3kd+2k+3d+1}{2d}$
$k < d$ and $k \nmid d$	$\frac{k^2+3kd+2k+3d+1}{2d} + (\lceil \frac{k+1}{d} \rceil)(k+1 \mod d)$
$k = d - 1$	$2k + 2$
Group-based KPS	
honest majority (per subgroup)	2

Fig. 10 Summary of communication costs

protocol is between $2(k+1)N$ messages when nodes can send authentication information directly to the intended recipient, and $\left(\frac{1}{2}(k^2 + 5k + 6) - 2\right) N$ messages when $d = 1$ and all authentication information must be routed hop-by-hop.

Our final protocol utilized shared group keys and protects against localized clusters of malicious nodes. The network is divided into subgroups, and the protocol can proceed as long as there is an honest majority of nodes in each subgroup. The communication overhead in this protocol is minimal, however it may require specialized tools to protect against certain attacks. Each node broadcasts its own reading, and later broadcasts the updated aggregate, but must be active to listen to and record the messages of other nodes within its subgroup. The communication complexity of this protocol during regular operation is therefore $2N$ messages.

Our focus was providing resilient data aggregation and early termination in the presence of an active adversary. Future work on aggregation in linear networks should focus on adapting protocols to other linear settings, such as thick linear networks, and determining if full private or concealed aggregation can be achieved in such a restricted problem setting without sacrificing early termination or requiring a significant increase in communication complexity.

Acknowledgements The authors would like to thank Konstantinos Patsakis for his comments on an earlier version. This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the discovery grant 203114-11.

References

1. Frederik Armknecht, Dirk Westhoff, Joao Girão, and Alban Hessler. A lifetime-optimized end-to-end encryption scheme for sensor networks allowing in-network processing. *Computer Communications*, 31(4):734–749, 2008.
2. Joshua Ash and Lee Potter. Sensor network localization via received signal strength measurements with directional antennas. *Proceedings of the Forty-Second Annual Allerton Conference on Communication, Control, and Computing*, pages 1861–1870, Champaign-Urbana , IL, Sep 2004.
3. Rabindra Bista and Jae-Woo Chang. Privacy-preserving data aggregation protocols for wireless sensor networks: A survey. *Sensors*, 10(5):4577–4601, 2010.
4. Levente Buttyán, Péter Schaffer, and István Vajda. Cora: Correlation-based resilient aggregation in sensor networks. *Ad Hoc Networks*, 7(6):1035–1050, 2009.
5. Claude Castelluccia, Aldar C-F. Chan, Einar Mykletun, and Gene Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5:20:1–20:36, June 2009.
6. Aldar C.-F. Chan and Claude Castelluccia. A security framework for privacy-preserving data aggregation in wireless sensor networks. *TOSN*, 7(4):29, 2011.
7. Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 278–287, New York, NY, USA, 2006. ACM.

8. Elena Fasolo, Michele Rossi, Jörg Widmer, and Michele Zorzi. In-network aggregation techniques for wireless sensor networks: a survey. *IEEE Wireless Commun.*, 14(2):70–87, 2007.
9. Keith B. Frikken and Joseph A. Dougherty, IV. An efficient integrity-preserving scheme for hierarchical sensor aggregation. In *Proceedings of the first ACM conference on Wireless network security*, WiSec '08, pages 68–76, New York, NY, USA, 2008. ACM.
10. J. Girao, D. Westhoff, and M. Schneider. CDA: concealed data aggregation for reverse multicast traffic in wireless sensor networks. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 5, pages 3044 – 3049 Vol. 5, may 2005.
11. Wenbo He, Xue Liu, Hoang Nguyen, Klara Nahrstedt, and Tarek F. Abdelzaher. PDA: Privacy-preserving data aggregation in wireless sensor networks. In *INFOCOM*, pages 2045–2053. IEEE, 2007.
12. Wenbo He, Hoang Nguyen, Xue Liu, Klara Nahrstedt, and Tarek Abdelzaher. iPDA: an Integrity-Protecting private data aggregation scheme for wireless sensor networks. In *Proceedings of the 2008 Military Communications Conference (MILCOM 2008)*, 2008.
13. Lingxuan Hu and David Evans. Secure aggregation for wireless network. In *SAINT Workshops*, pages 384–394. IEEE Computer Society, 2003.
14. Imad Jawhar and Nader Mohamed. A hierarchical and topological classification of linear sensor networks. In *Proceedings of the 2009 conference on Wireless Telecommunications Symposium*, WTS'09, pages 72–79, Piscataway, NJ, USA, 2009. IEEE Press.
15. Zhijun Li and Guang Gong. Data aggregation integrity based on homomorphic primitives in sensor networks. In *Proceedings of the 9th international conference on Ad-hoc, mobile and wireless networks*, ADHOC-NOW'10, pages 149–162, Berlin, Heidelberg, 2010. Springer-Verlag.
16. Mark Manulis and Jörg Schwenk. Security model and framework for information aggregation in sensor networks. *TOSN*, 5(2), 2009.
17. Keith M. Martin and Maura Paterson. An application-oriented framework for wireless sensor network key establishment. *Electron. Notes Theor. Comput. Sci.*, 192:31–41, May 2008.
18. Keith M. Martin and Maura B. Paterson. Ultra-lightweight key predistribution in wireless sensor networks for monitoring linear infrastructure. In *Proceedings of the 3rd IFIP WG 11.2 International Workshop on Information Security Theory and Practice. Smart Devices, Pervasive Systems, and Ubiquitous Networks*, WISTP '09, pages 143–152, Berlin, Heidelberg, 2009. Springer-Verlag.
19. Suat Ozdemir, Miao Peng, and Yang Xiao. PRDA: polynomial regression-based privacy-preserving data aggregation for wireless sensor networks. *Wireless Communications and Mobile Computing*, 2013.
20. Steffen Peter, Dirk Westhoff, and Claude Castelluccia. A survey on the encryption of convergecast traffic with in-network processing. *IEEE Trans. Dependable Sec. Comput.*, 7(1):20–34, 2010.
21. Bartosz Przydatek, Dawn Xiaodong Song, and Adrian Perrig. SIA: secure information aggregation in sensor networks. In Ian F. Akyildiz, Deborah Estrin, David E. Culler, and Mani B. Srivastava, editors, *SenSys*, pages 255–265. ACM, 2003.
22. Kasper Bonne Rasmussen and Srdjan Capkun. Implications of radio fingerprinting on the security of sensor networks. In *Proceedings of IEEE SecureComm*, pages 331 –340, sep. 2007.
23. Kasper Bonne Rasmussen and Srdjan Čapkun. Realization of RF distance bounding. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, pages 25–25, Berkeley, CA, USA, 2010. USENIX Association.
24. Sankardas Roy, Sanjeev Setia, and Sushil Jajodia. Attack-resilient hierarchical data aggregation in sensor networks. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, SASN '06, pages 71–82, New York, NY, USA, 2006. ACM.
25. David Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, SASN '04, pages 78–87, New York, NY, USA, 2004. ACM.
26. Geng Yang, Sen Li, Xiaolong Xu, Hua Dai, and Zhen Yang. Precision-enhanced and encryption-mixed privacy-preserving data aggregation in wireless sensor networks. *IJDSN*, 2013.