Secure Data Aggregation in Wireless Sensor Networks: Filtering out the Attacker's Impact

Sankardas Roy, Mauro Conti, Sanjeev Setia, and Sushil Jajodia

***Abstract*—Wireless sensor networks (WSNs) are increasingly used in several applications, such as volcano and fire monitoring, urban sensing, and perimeter surveillance. In a large WSN, *in-network data aggregation* (i.e., combining partial results at intermediate nodes during message routing) significantly reduces the amount of communication overhead and energy consumption. The research community proposed a loss-resilient aggregation framework called *synopsis diffusion* which uses duplicate-insensitive algorithms on top of multi-path routing schemes to accurately compute aggregates (e.g., predicate Count, Sum). However, this aggregation framework does not address the problem of false sub-aggregate values contributed by compromised nodes. This attack may cause large errors in the aggregate computed at the base station which is the root node in the aggregation hierarchy.**

In this paper, we make the synopsis diffusion approach secure against the above attack launched by the compromised nodes. In particular, we present an algorithm to enable the base station to securely compute predicate Count or Sum even in the presence of such an attack. Our *attack-resilient computation algorithm* computes the true aggregate by filtering out the contributions of compromised nodes in the aggregation hierarchy. Thorough theoretical analysis and extensive simulation study show that our algorithm outperforms other existing approaches.

***Index Terms*—Data Aggregation, Hierarchical Aggregation, In-network Aggregation, Sensor Network Security, Synopsis Diffusion, Attack-Resilient**

I. Introduction

Over the last decade WSNs are increasingly used in several real-world applications [1], [2], [3], [4], such as wild habitat monitoring, volcano and fire monitoring, urban sensing, and military surveillance. In most cases, the sensor nodes form a multi-hop network while the base station (BS) acts as the central point of control. Typically, a sensor node has limitation in terms of computation capability and energy reserves. The BS wants to collect the sensed information from the network. One common way is to allow each sensor node to forward its reading to the BS, possibly via other intermediate nodes. Finally, the BS processes the received data. However, this method is prohibitively expensive in terms of communication overhead.

In-network data aggregation [5], [6] can reduce the amount of communication and hence the energy consumed, especially in large WSNs. The main idea is to combine partial results at intermediate nodes during message routing. One approach [5], [6] is to construct a spanning tree rooted at the BS, and then perform in-network aggregation along the tree. The important aggregates considered by the research community include Count, and Sum. It is straightforward to generalize these aggregates to predicate Count (e.g., number of sensors whose reading is higher than 10 unit) and Sum. In addition, Average can be computed from Count and Sum. We can also easily extend a Sum algorithm to compute Standard Deviation and Statistical Moment of any order.

However, communication losses resulting from node and transmission failures, which are common in WSNs, can adversely affect tree-based aggregation approaches. To address this problem, we can make use of multi-path routing techniques for forwarding sub-aggregates [5]. For duplicate-insensitive aggregates such as Min and Max, this approach provides a fault-tolerant solution. Unfortunately, for duplicate-sensitive aggregates, such as Count and Sum, multi-path routing leads to double-counting of sensor readings. Recently, several researchers [7], [8] have presented clever algorithms to solve this double-counting problem. A robust and scalable aggregation framework called *synopsis diffusion* has been proposed for computing duplicate-sensitive aggregates, such as Count and Sum. This approach uses a ring topology where a node may have multiple parents in the aggregation hierarchy. Furthermore, each sensed value or sub-aggregate is represented by a duplicate-insensitive bitmap called *synopsis*.

The possibility of node compromise introduces more challenges because most of the existing in-network aggregation algorithms have no provisions for security. A compromised node might attempt to thwart the aggregation process by launching several attacks, such as eavesdropping, jamming, message dropping, message fabrication, and so on. This paper focuses on a subclass of these attacks in which the adversary aims to cause the BS to derive an incorrect aggregate. By relaying a false sub-aggregate to the parent node, a compromised node may contribute a large amount of error to the aggregate. As an example, during the Sum computation algorithm [8], [7], a compromised node X can inject an arbitrary amount of error in the final estimate of Sum by falsifying X's own sub-aggregate. We refer to this attack as the *falsified sub-aggregate attack*. The threat model is detailed in Section IV.

In this paper, we design an algorithm to securely compute

S. Roy is with the Computing and Information Sciences Department, Kansas State University, USA.

M. Conti is with the Department of Mathematics in University of Padua, Italy, and with the Center for Secure Information Systems in George Mason University, Fairfax, VA, USA. He is supported by a Marie Curie Fellowship funded by the European Commission for the PRISM-CODE project (Privacy and Security for Mobile Cooperative Devices) under the agreement n. PCIG11-GA-2012-321980. This work has been partially supported by the TENACE PRIN Project 20103P34XC funded by the Italian MIUR.

S. Setia is with the Department of Computer Science in George Mason University, Fairfax, VA, USA.

S. Jajodia is with the Center for Secure Information Systems in George Mason University, Fairfax, VA, USA.

aggregates, such as Count and Sum despite the falsified sub-aggregate attack. In particular, our algorithm which we call the *attack-resilient computation algorithm* consists of two phases. The main idea is as follows: (i) In the first phase, the BS derives a preliminary estimate of the aggregate based on minimal authentication information received from the nodes. (ii) In the second phase, the BS demands more authentication information from only a subset of nodes while this subset is determined by the estimate of the first phase. Finally, the BS is able to filter out the false contributions of the compromised nodes from the aggregate. The key observation which we exploit to minimize the communication overhead is that to verify the correctness of the final synopsis (representing the aggregate of the whole network) the BS does not need to receive authentication messages from all of the nodes.

We examine the performance of our algorithm via both thorough theoretical analysis and extensive simulation. The per-node overall communication overhead in our algorithm is $max(O(m\log A), O(mt))$ where m is $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$, A is the aggregate value, and t compromised nodes are present in the network. Note that m items are computed in parallel fashion to result in an (ε,δ)-approximate aggregate as discussed in [9]. When Count is computed, $A = N$ where N is the total number of nodes in the network; when Sum is computed, $O(\log A) = O(\log N) + O(\log v)$ where v is a single node's maximum value. An existing attack-resilient algorithm [10] incurs $O(N)$ communication overhead in the worst case, which is much higher than ours given $t << N$, and the unit of sensed values are such that $\log(v) << N$. Furthermore, our algorithm incurs $O(1)$ latency while the other existing algorithm [11] takes $O(\log N)$ latency whereas both the algorithms (i.e. ours and [11]) essentially incur the same communication overhead to ensure the same approximation error guarantee.

One may be interested in knowing the difference of our current work from our previously published related work. Although our prior work [12] considers the same aggregation framework (i.e. synopsis diffusion) and similar attack scenario (i.e. falsified sub-aggregate attack) the goal (as well as the outcome) is very different. Our prior work [12] presents only a verification algorithm, which would fail to compute the aggregate in the presence of an attack while our current work will be successful in doing so. Furthermore, we previously presented another attack-resilient aggregation algorithm [13] for the synopsis diffusion framework, but the algorithm proposed in the current paper is more efficient. We stress that the design principle of our current approach is much different from that in [13], e.g., using a preliminary round to derive an estimate of the aggregate which helps reduce the communication overhead of the whole process is absent in [13]. Moreover, the theoretical analysis technique present in the current work is also different from that in [13] or [12].

It is to be noted that while our algorithm is designed having WSNs in mind, it is straightforward to extend our solution for secure aggregation query processing in a large-scale distributed system such as a distributed database system over the Internet [9].

The rest of this paper is organized as follows. Section II reviews the body of related work, and Section III briefly presents the synopsis diffusion approach. Section IV describes the problem statement, the assumptions, and the threat model. Section V describes our attack-resilient protocol. Section VI presents the simulation results, and Section VII concludes this paper. The Appendix section presents some additional results.

II. Related Work

Several researchers have studied problems related to data aggregation in WSNs.

A. Data Aggregation in a Trusted Environment

The Tiny Aggregation Service (TAG) to compute aggregates, such as Count and Sum, using tree-based aggregation algorithms were proposed in [5]. Similar algorithms to compute Count and Sum were proposed in [6]. Moreover, tree-based aggregation algorithms to compute an order-statistic (i.e., quantile) have been proposed in [14].

To address the communication loss problem in tree-based algorithms an aggregation framework called *synopsis diffusion* is designed in [8], which computes Count and Sum using a ring topology. Very similar algorithms are independently proposed in [7]. These works use duplicate-insensitive algorithms for computing aggregates based on [15]'s algorithm for counting distinct elements in a multi-set.

TABLE I
Comparing Our Work with the Prior Schemes

Algori -thms	Aggregates considered	Number of com--promised nodes	Verification	Attack-resilient compu-tation
[5]	Count, Sum	0	None	None
[8], [7]	Count, Sum	0	None	None
[16]	Count, Sum	1	Count, Sum	None
[17]	Count, Sum	≥ 1	Count, Sum	None
[9]	Count, Sum	≥ 1	Count, Sum	None
[12]	Count, Sum	≥ 1	Count, Sum	None
[10]	Count, Sum	≥ 1	Count, Sum	Count, Sum
[11]	Count, Sum	≥ 1	Count, Sum	Count, Sum
[13]	Count, Sum	≥ 1	Count, Sum	Count, Sum
Our work	Count, Sum	≥ 1	Count, Sum	Count, Sum

B. Secure Aggregation Techniques

Several secure aggregation algorithms have been proposed assuming that the BS is the only aggregator node in the network [18], [19], [20]. These works did not consider in-network aggregation. Only recently, the research community has been paying attention to the security issues of hierarchical aggregation.

The first attack-resilient hierarchical data aggregation protocol was designed in [16]. However, this scheme is secure when only one malicious nodes is present. A tree-based verification algorithm was designed in [17], [21] by which the BS can detect if the final aggregate, Count or Sum, is falsified. A few verification algorithms for computing Count and Sum within the synopsis diffusion approach were designed in [9], [12]. Recently, a few novel protocols have been proposed for

'secure outsourced aggregation' [22]; however, as noted by the authors, these algorithms are not designed for WSNs.

Although [17], [21], [9] prevent the BS from accepting a false aggregate, they do not guarantee the successful computation of the aggregate in the presence of the attack. We further stress that our own prior work [12] presents only a verification algorithm for the synopsis diffusion framework, which would fail in the presence of an attack. The attestation phase of SDAP [10] can be expensively used to compute Count and Sum in the presence of a few compromised nodes. Recently, an attack-resilient aggregation algorithm for computing Count and Sum has been proposed in [11], which is based on a sampling technique. Despite the adversarial interference, this algorithm can produce a $(\varepsilon,\delta)-$approximation of the target aggregate. We previously presented an attack-resilient aggregation algorithm [13] for the synopsis diffusion framework, but the current attack-resilient algorithm proposed in this paper is more efficient. We thoroughly compare our current work with all the prior attack-resilient algorithms [13], [10], [11] in Section V-F.

We compare the security features of our schemes with those of the existing schemes in Table I. For each scheme, we indicate how many compromised nodes against which it is secure. We also state if the scheme has provisions to verify the computed aggregate and to compute the aggregate in the presence of compromised nodes.

III. Preliminaries: Synopsis Diffusion

In [8] and [7], the authors proposed an aggregation framework called *synopsis diffusion* which is based on a ring topology as illustrated in Figure 1. During the query distribution phase, nodes form a set of rings around the base station (BS) based on their distance in terms of hops from BS. By T_i we denote the ring consisting of the nodes which are i hops away from BS. In the subsequent aggregation period, starting in the outermost ring, each node generates and broadcasts a local synopsis. The synopsis generation function is represented by $SG(v)$, where v is the sensor value relevant to the query. A node in ring T_i will receive broadcasts from all of the nodes in its communication range in ring T_{i+1}. It will then combine its own local synopsis with the synopses received from its children using a synopsis fusion function $SF()$ and then broadcast the updated synopsis. Thus, the fused synopses propagate level-by-level until they reach BS, which combines the received synopses using $SF()$. Finally, BS uses the synopsis evaluation function $SE()$ to translate the final synopsis to the answer to the query. We now describe the synopsis diffusion algorithms for Count and Sum. These algorithms are based on Flajolet and Martin's probabilistic algorithm for counting the number of distinct elements in a multi-set [15]. Each node X generates a local synopsis Q^X which is a bit vector. We now present a definition which we repeatedly use in this paper.

Definition: A node X's *fused synopsis*, B^X, is recursively defined as follows. If X is a leaf node (i.e., X is in the outermost ring), B^X is its local synopsis Q^X. If X is a non-leaf node, B^X is the logical OR of X's local synopsis Q^X with X's children's *fused synopses*.

If node X receives synopses $B^{X_1}, B^{X_2}, \ldots, B^{X_d}$ from d child nodes X_1, X_2, ..., X_d, respectively, then X computes B^X as follows:

$$B^X = Q^X || B^{X_1} || B^{X_2} || \ldots || B^{X_d}, \quad (1)$$

where $||$ denotes the bitwise OR operator. Note that B^X represents the sub-aggregate of node X, including its descendant nodes. We note that B^{BS} is same as the final synopsis B.

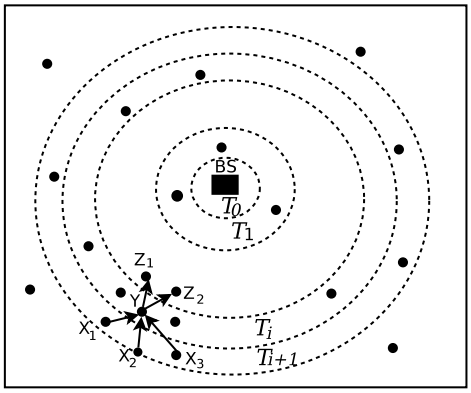

Fig. 1. Synopsis Diffusion over a Ring Topology—A node may have multiple parents and multiple children. E.g. Node Y in ring T_i has 2 parents, Z_1, and Z_2 of ring T_{i-1}. Y has 3 child nodes, X_1, X_2, and X_3 of ring T_{i+1}. For brevity all rings are not shown. The base station (BS) is at ring 0.

A. Count

In this algorithm, each node X generates a local synopsis Q^X which is a bit vector of length $\eta > \log N'$, where N' is the upper bound on Count. To generate Q^X, node X executes the function $CoinToss(X,\eta)$ given below (Algorithm 1), where X is the node's identifier. Algorithm 1 can be interpreted as a coin-tossing experiment with a hash function. The function $hashOf()$ whose output is 0 or 1 simulates a fair coin-toss. $CoinToss(X,\eta)$ returns the number of attempts, say i, until the first head occurs or $\eta+1$ if all of η tosses have been tail. In the synopsis generation function SG_{count}, the i-th bit of Q^X is set to '1' while all other bits are '0'. Thus, Q^X is a bit vector of the form $0^{(i-1)}.1.0^{(\eta-i)}$ with probability 2^{-i}.

```
begin
    attemptNum=1;
    while attemptNum < η+1 AND hashOf(<X,attemptNum>) = 0 do
        attemptNum=attemptNum+1;
    end
    return attemptNum;
end
```

Algorithm 1: $CoinToss(X,\eta)$ as executed by node X

The synopsis fusion function $SF()$ is the bitwise Boolean OR. Each node X fuses its local synopsis Q^X with the synopses it receives from its children. Let B denote the final synopsis computed by BS. We observe that B will be a bit vector of length η of the form $1^{z-1}.0.[0,1]^{\eta-z}$, where z is the lowest-order bit in B that is 0. BS can estimate Count from B via the synopsis evaluation function $SE()$: The count of nodes in the network is $2^{z-1}/0.7735$. The synopsis evaluation function $SE()$ is based on Property 2 below. Intuitively, the number of

sensor nodes is proportional to 2^{z-1} since no node has set the z-th bit while computing $CoinToss(X,\eta)$.

Below we present a few important properties of the final synopsis B computed at BS. The first three properties have been derived in [15], [7], while Property 4 is documented from our observation. Let $B[i], 1 \leq i \leq \eta$ denote the i-th bit of B, where bits are numbered starting from the left. Also, N is the number of nodes present in the network.

Property 1. For $i < \log_2 N - 2\log_2\log_2 N$, $B[i] = 1$ with probability ≈ 1. For $i \geq \frac{3}{2}\log_2 N$, $B[i] = 0$ with probability ≈ 1.

This result implies that for a network of N nodes, we expect that B has an initial prefix of all ones and a suffix of all zeros, while only the bits around $B[\log_2 N]$ exhibit much variation. This provides an estimate of the number of bits, η, required for a node's local synopsis. In practice, $\eta = \log_2 N' + 4$ bits are sufficient to represent B with high probability [15], where N' is the upper bound of Count. This result also indicates that the length of the prefix of all ones in B can be used to estimate N. Let $z = \min\{i|B[i] = 0\}$, i.e., z is the location of the leftmost zero in B. Then $R = z-1$ is a random variable representing the length of the prefix of all ones in the synopsis. The following results hold for R.

Property 2. The expected value of R, $E(R) \approx \log_2(\phi N)$, where the constant ϕ is approximately 0.7735.

This result implies that R can be used as an unbiased estimator of $\log_2(\phi N)$. It is the basis for the synopsis evaluation function $SE()$, which estimates N as $2^R/\phi$.

Property 3. The standard deviation of $R, \sigma_R \approx 1.1213$.

To reduce the standard deviation of R, [15] proposed an algorithm named PCSA, where m synopses are computed in parallel.

Property 4. If N nodes participate in Count algorithm, the expected number of nodes that will contribute a '1' to the i-th bit of the final synopsis B is $N/2^i$.

We call these nodes *contributing nodes* for bit i of B. This property is derived from the observation that each node X sets the i-th bit of its local synopsis Q^X with probability 2^{-i}. As an example, for bit $r = E(R) = \log_2(\phi N)$, the expected number of contributing nodes is $1/\phi \approx 1.29$. This result also implies that the expected number of nodes that contribute a '1' to the bits right to the i-th bit (i.e., bits j, where $i < j \leq \eta$) is approximately $N/2^i$. As an example, the expected number of contributing nodes for bits $j \geq r+1$ is approximately $1/\phi$.

B. Sum

The Count algorithm can be extended for computing Sum. The synopsis generation function $SG()$ for Sum is a modification of that for Count, while the fusion function $SF()$ and the evaluation function $SE()$ for Sum are identical to those for Count.

To generate the local synopsis Q^X to represent its sensed value v_X, node X invokes the function $CoinToss()$, used for Count synopsis generation, v_X times[1]. In the i-th invocation ($1 \leq i \leq v_X$), node X executes the function $CoinToss(X_i,\eta)$ where key_i is constructed by concatenating its ID and integer i (i.e., $key_i = < X, i >$), and η is the synopsis length. The value of η is taken as $\log_2 S' + 4$, where S' is an upper bound on the value of Sum aggregate. Note that unlike the local synopsis of a node for Count, more than one bit in the local synopsis of a node for Sum may be equal to '1'. The pseudo code of the synopsis generation function, $SG_{sum}(X, v_X, \eta)$, is presented below (Algorithm 2).

```
begin
    Q^X[index] = 0  ∀index,  1 ≤ index ≤ η;
    i=1;
    while i ≤ v_X do
        key_i = < X, i >;
        index = CoinToss(key_i, η);
        Q^X[index] = 1;
        i = i + 1;
    end
    return Q^X;
end
```

Algorithm 2: $SG_{sum}(X, v_X, \eta)$ as executed by node X

Note that Count can be considered as a special case of Sum where each node's sensor reading is equal to one unit. [7] showed that Properties 1, 2, 3, and 4 described above for Count synopsis also hold for Sum synopsis, with appropriate modifications. Below we present these properties of Sum synopsis, which we will find useful in the rest of this paper. Let $B[i], 1 \leq i \leq \eta$ denote the i-th bit of the final synopsis B, where bits are numbered starting from the left. Furthermore, S is the Sum of the sensed values of the nodes present in the network.

Property 1. For $i < \log_2 S - 2\log_2\log_2 S$, $B[i] = 1$ with probability ≈ 1. For $i \geq \frac{3}{2}\log_2 S$, $B[i] = 0$ with probability ≈ 1.

Property 2. Let R represent the length of the prefix of all ones in B, i.e., $R = z-1$ where $z = \min\{i|B[i] = 0\}$. The expected value of R, $E(R) \approx \log_2(\phi S)$, where the constant ϕ is approximately 0.7735.

Property 3. The standard deviation of $R, \sigma_R \approx 1.1213$.

Unlike the above properties, Property 4 is not a straightforward extension of its counterpart for Count synopsis. From the construction of the synopsis generation function, $SG_{sum}()$ (Algorithm 2), we observe that if the Sum is S, then the function $CoinToss()$ is invoked S times in total considering synopsis generation of all nodes. Each node X gets a chance to set the i-th bit of Q^X, its local synopsis, v_X times—each time with probability 2^{-i}. So, the expected number of contributing nodes for the i-th bit of B not only depends on the total number of nodes N and the value of i but also on the distribution of sensor readings.

Property 4. The expected number of invocations of $CoinToss()$ that will contribute a '1' to the i-th bit of the final synopsis B is $S/2^i$, where S is the value of Sum.

As an example, with $r = E(R) = \log_2(\phi S)$, the expected number of invocations of $CoinToss()$ which set the r-th to '1' is $1/\phi \approx 1.29$. This result also implies that the expected number of contributing nodes for bit r is less than $1/\phi$. Furthermore, the expected number of invocations of $CoinToss()$ that contribute a '1' to the bits right to the i-th bit (i.e., bits j, where $i < j \leq \eta$) is approximately $S/2^i$. As an example, the expected number of invocations of $CoinToss()$ that contribute a '1' to the bits right to the r-th bit is approximately $1/\phi$,

[1] Without loss of generality, each sensor reading is assumed to be an integer.

which implies that the expected number of contributing nodes for the bits to the right of the r-th bit is less than $1/\phi$.

Similarly, as in the case of Count, the PCSA algorithm can be used to reduce the error in the estimate for Sum by computing m synopses in parallel.

IV. The Problem Statement

We now present the assumptions, discuss the threat model, and formally state the problem that we address in this paper.

A. Assumptions

We assume that sensor nodes are similar to the current generation of sensor nodes, e.g., MicaZ or Telos motes, in their computational and communication capabilities and power resources, while BS is a laptop class device supplied with long-lasting power. We assume that BS cannot be compromised and it uses a protocol such as μTesla [23] to authenticate its broadcast messages to the network nodes. We also assume each node shares a pair-wise key with BS. Let the key of the node with ID X be denoted as K_X. To authenticate a message to BS, a node X sends a MAC (Message Authentication Code) generated using the key K_X. We further assume that each pair of neighboring nodes has a pairwise key to authenticate its mutual communication.

B. Threat Model

The synopsis diffusion framework on its own does not include any provisions for security. To stop unauthorized nodes from interfering in (or eavesdropping on) communications among honest nodes, we can extend the aggregation framework with standard authentication and encryption protocols. So, we do not see any need to consider the attacks coming from unauthorized nodes in the rest of this paper. However, cryptographic mechanisms cannot prevent attacks launched by compromised nodes because the adversary can obtain cryptographic keys from the compromised nodes. Compromised nodes might attempt to thwart the aggregate computation process in multiple ways. A compromised node C which happens to be an in-network data aggregator may leak (to the adversary) the sensor readings (and sub-aggregates) which C receives from C's child nodes. Several researchers [24] already proposed privacy-preserving aggregation algorithms, and we do not consider this problem in the rest of this paper. Below we discuss other potential problems and identify the scope of this paper.

1. Falsifying the local value: A compromised node C can falsify its own sensor reading with the goal of influencing the aggregate value. There are three cases. **Case (i)**: If the local value of a honest node can be *any* value (i.e. not bounded by the domain of application), then a compromised node can pretend to sense *any* value. In this case, there is no way to detect the falsified local value attack (as also confirmed in [17]). We leave Case (i) out of the scope of this paper. **Case (ii)**: If the local value of a honest node is bounded, and a compromised node falsifies the local value within the bound, there is no solution for detecting such an attack as in Case (i). We only observe that in Case (ii), the impact of this attack is limited as explained in the Appendix. **Case (iii)**: The local value of a honest node is bounded, and a compromised node falsifies the local value outside the bound. Our proposed algorithm does detect and guard against Case (iii) attack scenario as discussed in Section V-D.
2. Falsifying the sub-aggregate: A compromised node C can falsify the sub-aggregate which C is supposed to compute based on the messages received from C's child nodes. It is challenging to guard against this attack, and addressing this challenge is the main focus of this paper.

We assume that if a node is compromised, all the information it holds will be compromised. We conservatively assume that all malicious nodes can collude or can be under the control of a single attacker. We use a Byzantine fault model, where the adversary can inject any message through the compromised nodes. Compromised nodes may behave in arbitrarily malicious ways, which means that the *sub-aggregate* of a compromised node can be arbitrarily generated.

C. The Goal of the Paper

Our goal is to enable BS to obtain the 'true' estimate of the aggregate (which BS would compute if there were no compromised nodes) even in the presence of the attack. More formally, goal (a) is to detect if $\hat{B}$, the synopsis received at BS is the same as the 'true' final synopsis B, and goal (b) is to compute B from $\hat{B}$ and other received information. Without loss of generality, we present our algorithms in the context of Sum aggregate. As Count is a special case of Sum, where each node reports a unit value, these algorithms are readily applicable to Count aggregate also.

D. The Attack Details

Note that BS estimates the aggregate based on the lowest-order bit z that is '0' in the final synopsis. So, a compromised node C attempts to falsify its fused synopsis B^C such that it would affect the value of z. Node C simply inserts '1's in one or more bits in positions j, where $z \leq j \leq \eta$, into B^C which C broadcasts to its parents. Let $\hat{B^C}$ denote the synopsis finally broadcast by node C. Note that the compromised node C does not need to know the true value of z; it can simply set some higher-order bits to '1' with the expectation that this will affect the value of z computed by BS.

Since the synopsis fusion function is a bitwise Boolean OR, the fused synopsis computed at any node which is at the higher level than node C on the aggregation hierarchy will contain the false contributions of node C. We observe that when a node X computes the fused synopsis $\hat{B^X}$, X is not sure if $\hat{B^X}$ contains any false '1's contributed by a compromised node lower in the hierarchy. The observation is true also for the BS when it computes the final synopsis $\hat{B}$. We call the '1' bits which are present in $\hat{B}$ but not in B as *false '1's* in the rest of this paper.

Note that a compromised node C can introduce a false '1' at bit j in $\hat{B^C}$ by launching either of the following attacks.

- Falsified sub-aggregate attack: C just flips bit j in $\hat{B^C}$ from '0' to '1'—not having a local aggregate justifying that '1' in the synopsis $\hat{B^C}$.
- Falsified local value attack: C injects a false '1' at bit j in its local synopsis, Q^C. The falsified synopsis, $\hat{Q^C}$ induces bit j in $\hat{B^C}$ to be '1'. Note that true local sensed value, v_C corresponds to Q^C.

An example of the falsified sub-aggregate attack is shown in Figure 2. Node P is compromised i.e., it is an example of node C in the above discussion. It has three child nodes which are X, Y and Z, and P receives from them synopses $\hat{B^X}$, $\hat{B^Y}$, and $\hat{B^Z}$, respectively. Node P is supposed to aggregate its local synopsis Q^P with the received synopses using the boolean OR operation. That means, the fused synopsis of P should be $\hat{B^P} = Q^P||\hat{B^X}||\hat{B^Y}||\hat{B^Z}$. However, in this example, malicious node P increases the number of '1's in $\hat{B^P}$ by injecting false '1's into $\hat{B^P}$ without forging Q^P. The fabricated $\hat{B^P}$ represents a bogus sub-aggregate at P, which is higher than P's true sub-aggregate. Note that in another example (not shown in the figure), P could launch falsified local value attack by adding false '1's in Q^P.

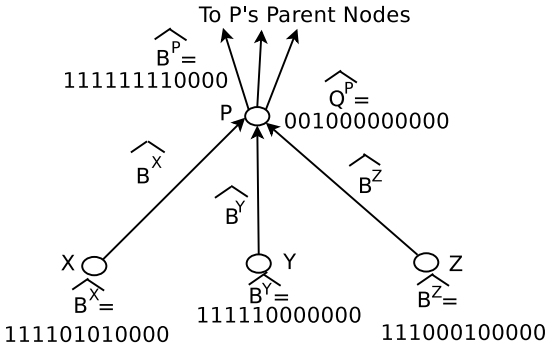

Fig. 2. Falsified Sub-aggregate Attack: Node P is supposed to aggregate its local synopsis Q^P with the received synopses (from the child nodes X, Y, and Z) using the boolean OR operation. However, being malicious node P injects false '1's in its fused synopsis $\hat{B^P}$. The fabricated $\hat{B^P}$ represents a bogus sub-aggregate at P, which can be higher than P's true sub-aggregate.

Let $\hat{R} = \hat{z} - 1$, where $\hat{z}$ be the lowest-order bit that is '0' in the received final synopsis $\hat{B}$. Also, let $R = z - 1$, where z is the lowest-order bit that is '0' in the correct final synopsis B. Then BS's estimate of the aggregate will be larger than the correct estimate by a factor of $2^{\hat{R}-R}$. So, a large amount of error will appear in the final estimate of BS.

We also observe that even a single node can launch this attack with a high rate of success because the use of multi-path routing in the synopsis diffusion approach makes it highly likely that the falsified synopsis will be propagated to BS. On the other hand, it is very hard to launch an attack which results in the aggregate estimated at BS being lower than the true estimate. This is because a compromised node C's changing bit j in its fused synopsis, B^C from '1' to '0' has no effect if there is another node X that contributes a '1' to bit j in its local synopsis Q^X and hence to bit j in the final synopsis B. To make this attack a success, the attacker must compromise all of the possible paths from node X to BS so that X's '1' cannot reach BS, which is hard to achieve. If there is more than one node which contributes to the same bit, then it is even harder.

In the rest of this paper, we do not further discuss the second type of attack (changing bit '1' to '0'). We restrict our discussion to the first type of attack (changing bit '0' to '1'), which we call the *false '1' injection attack*. That means the goal of our attacker is only to increase the estimate of the aggregate.

TABLE II
NOTATIONS USED IN DESCRIBING THE SECURE SUM PROTOCOLS

Symbol	Meaning
N	the total number of nodes
v_X	sensed value of node X
S	the value of Sum aggregate
K_X	symmetric key shared between node X and the BS
$MAC(K_X, M)$	message authentication code of message M computed using key K_X
$X \rightarrow Y$	X sends a message to Y
$X \rightarrow *$	X broadcasts a message to one hop neighbors
$X \rightarrow\rightarrow *$	X broadcasts a message to the network
$< a_1 \ , \ a_2 >$	concatenation of string a_1 and a_2
$\|\|$	the bitwise OR operator
t	number of compromised nodes
η	the length of the synopsis
Q^X	the local synopsis of node X
B^X	the fused synopsis of node X if no attack is in the network
$\hat{B^X}$	the fused synopsis actually computed by node X
B	the final synopsis at BS if no attack is in the network
$\hat{B}$	the final synopsis actually computed by BS

Notation A list of notations used in this paper is given in Table II.

V. COMPUTING COUNT AND SUM DESPITE ATTACKS

In this section, we propose an attack-resilient protocol which enables BS to compute the aggregate despite the presence of the attack.

A. Introducing Authentication Mechanisms

Let us remind the reader that a compromised node X launches the falsified sub-aggregate attack by inserting one or more *false '1's* in its fused synopsis. An obvious solution to guard against this attack is as follows. BS broadcasts an aggregation query message which includes a random value, *Seed*, associated to the current query. In the subsequent aggregation phase, along with the fused synopsis $\hat{B^X}$, each node X also sends a MAC to BS authenticating its sensed value v_X. Node X uses *Seed* and its own ID to compute its MAC. As a result, BS is able to detect and filter out any false '1' bits inserted in the final synopsis B.

Specifically, if node X contributes to bits $b_1, b_2, \ldots, b_\zeta$ in its local synopsis Q^X, it generates a MAC, $M = MAC(K_X, L)$, where K_X is the key that node X shares with BS and the content of L is $< X, v_X, b_1, b_2, \ldots, b_\zeta, Seed >$. Each node X sends a message (L', M) where $L' =< X, v_X, b_1, b_2, \ldots, b_\zeta >$ might be needed by BS to regenerate the MAC for the verification. We observe that this approach is not suitable for a WSN as it requires $O(N)$ MACs to be forwarded to BS. Our attack-resilient algorithm presented below also uses similar

MACs but reduces the total number of them. Throughout this paper when we say a message contains a MAC, M, it is understood that the corresponding L' is attached to M. To save space, we do not always explicitly mention this although we take into account the extra byte overhead in the simulation experiments.

Finally, in the rest of the paper, by the term *false MAC* we refer to any string that does not correspond to the MAC generation scheme described above. Note that a false MAC can be associated either to a false '1' or to a non-false '1' bit. Specifically, a compromised node X can generate a false MAC (in the context of computing the function $MAC(K_X, L)$) in four ways—(i) by using a false L, (ii) by using a false key K_X, (iii) by doing both of (i) and (ii) above, or (iv) by simply sending a bogus array of bits. As BS re-executes the MAC generation process for each received MAC, any false MAC will be detected by BS.

We observe that, in general, BS can verify the final synopsis if it receives one valid MAC for each '1' bit in the synopsis. In fact, to verify a particular '1' bit, say bit i, BS does not need to receive authentication messages from all of the nodes which contribute to bit i. As an example, more than half of the nodes are likely to contribute to the leftmost bit of the synopsis (Property 4 of Sum synopsis), while to verify this bit, BS needs to receive a MAC only from one of these nodes.

We now introduce the following notations. M_i^X denotes the MAC, generated by node X, authenticating the i-th bit of its local synopsis Q^X. Note that M_i^X is required to be generated only if $Q^X[i] = 1$, i.e. there are no MAC for '0' bits. Furthermore, for a particular i, M_i denotes one arbitrary element of the following set: $\{M_i^X \mid Q^X[i] = 1\}$, where elements of the set are enumerated with respect to X. As an example, if two nodes X_1 and X_2 set bit i to be '1' in their local synopses, then M_i corresponds to either $M_i^{X_1}$ or $M_i^{X_2}$.

We assume that a node X's message to one of its parents, P, can be lost due to communication failure but it cannot be partially or wrongly received—node-to-node authentication and acknowledgement mechanisms can be used to enforce this property. It implies that if B^X reaches P, all of the MACs sent by X also reaches P.

B. The Main Idea of the Protocol

If BS receives one valid MAC from a source node for each '1' bit in the final synopsis, it is able to correctly compute the aggregate. Before presenting our protocol, we describe a simpler protocol: Each node X forwards one MAC for each of the '1' bits in $\hat{B^X}$, and BS will verify all of the '1's in the received final synopsis $\hat{B}$. This protocol has the following flaw.

Let there be a compromised node C which falsely injects a few '1's in its fused synopsis $\hat{B^C}$ and sends a false MAC for each of these false '1' bit. Then, with some probability, these false MACs may get selected at each hop before reaching BS. If for a bit in final synopsis B, say bit i, BS does not receive a valid MAC but only false MACs, then BS cannot determine the real state of bit i. In fact, this can be the consequence of either of the following two scenarios: (i) $B[i] = 0$ and a false MAC has been generated; (ii) a source node (possibly a few hops away from BS) has sent a valid MAC for bit i ($B[i] = 1$, indeed), but this MAC lost the race to false MACs in the random selection procedure en-route BS.

However, we observe that the probability of this 'undecidability' problem to arise is not the same for all of the bits. In fact, a false MAC is not equally likely to get selected for all of the bits because the number of source nodes that contribute to a bit (hence, the number of valid MACs) varies with the bit position. Property 4 of Sum synopsis (discussed in Section III-B) says that the number of source nodes increases exponentially from the right to the left. As an example, approximately, $1/\phi = 1.27$ nodes are expected to contribute to bit r, $2/\phi = 2.54$ nodes are expected to contribute to bit $(r-1)$, and so on, where r is the expected length of the prefix of consecutive '1's in the final synopsis B. So, if the number of compromised nodes, t, is small compared to the total number of nodes, N, we expect that BS will receive a valid MAC for the left bits far from bit r, but may not receive a valid MAC for the other bits.

Considering the above observation, we design an attack-resilient protocol having two phases as follows:

- In phase one, we run the simple protocol described above. That is, each node X forwards one randomly selected MAC for each '1' bit in $\hat{B^X}$. At the end of this phase, BS verifies the received MACs. The '1's in $\hat{B}$ for which no valid MACs have been received by BS are reset to '0'. Let $\bar{B}$ represent the final synopsis at BS after the above filtering process is performed. Analyzing $\bar{B}$, we make an estimate, $\hat{r}$ of the expected prefix length, r of B. We will show later that $\hat{r}$ is a lower bound of r.
- In phase two, nodes which contribute to bit $\hat{r}$ or to the bits to the right of bit $\hat{r}$ send a MAC to BS. In this phase, no random selection technique is employed in forwarding MACs—each node forwards all of the received MACs toward BS.

We will show later that, given a deviation of $\hat{r}$ from r to the left, the number of MACs (hence the communication overhead) required in the second phase is exponential of this deviation. The main challenge is how to get a good estimate, $\hat{r}$ in the first phase. We will show that in the presence of t compromised nodes, the deviation can be kept within $O(\log_2 t)$. In this case, the number of MACs transmitted (per synopsis) in the second phase will be $O(t)$, i.e. proportional to the number of compromised nodes, t.

C. The Operation Details of the Protocol

We stress that like the original synopsis diffusion approach our attack-resilient protocol computes multiple synopses (say m) in parallel. Multiple synopses are computed to reduce the approximation error in the estimate of the aggregate [7], [8]. However, for the sake of clarity, we below present our protocol only considering a single synopsis, which should be executed in the same way for each synopsis. Table III introduces a few additional notations which we use to describe this protocol.

As in the original synopsis diffusion algorithm, during the query distribution phase nodes arrange themselves into a ring

topology around BS. Upon receiving the query message, the nodes aggregate their local synopses with their child nodes' synopses and send some authentication messages to BS in the following two phases.

TABLE III
NOTATIONS TO DESCRIBE THE ATTACK-RESILIENT PROTOCOL.

Symbol	Meaning
B	the final synopsis if no attack is launched
B_i	the i-th bit of synopsis B
$B[i]$	the value of the i-th bit of synopsis B
R	the length of the prefix of all '1's in B
r	the expected value of R
$\hat{r}$	BS's estimate of r after phase one
$\hat{B}$	the final synopsis received by BS in phase one in the presence of false MAC injection attack
$\hat{R}$	the length of the prefix of all '1's in $\hat{B}$
$\bar{B}$	the final synopsis at BS after false MACs, if any are filtered in phase one
$\bar{R}$	the length of the prefix of all '1's in $\bar{B}$
B'	the final synopsis at BS after phase two terminates
R'	the length of the prefix of all '1's in B'

1) Phase One: First, BS broadcasts a query message as follows:

$$BS \rightarrow\rightarrow * : \langle \text{“}PhaseOne\text{”}, \text{”}Sum\text{”}, Seed, \eta \rangle,$$

where “*PhaseOne*” is a flag indicating that phase one is going to begin, and η is the synopsis length.

In this phase, nodes basically execute the original synopsis diffusion algorithm with additional transmission of some MACs. In particular, each node X randomly selects one MAC for each '1' bit in synopses $\hat{B^X}$ from the MACs received from its child nodes (possibly including X's own MAC). X forwards the selected MACs to its parents. The message broadcast by X to its parent nodes is as follows:

$$X \rightarrow * : \langle\ \hat{B^X},\ \{M_i \mid \hat{B^X}[i] = 1, 1 \leq i \leq \eta\}\ \rangle,$$

where $\hat{B^X}$ represents the fused synopsis at node X, M_i represents a MAC corresponding to $\hat{B^X}[i]$.

We require a restriction on the number of MACs that a node can forward. In fact, if node X sends an aggregation message (synopses $\hat{B^X}$ and corresponding MACs) to its parent node Y, Y does not accept more than one MAC for each '1' bit in B^X. This assumption can be enforced by employing authentication techniques in the communication procedure among neighboring nodes.

After all of the MACs have been received by BS, for any '1' bit, say bit $\hat{B}_i$, in the synopses $\hat{B}$ for which no valid MAC has been received, BS resets $\hat{B}_i$ to '0'. The resulting set of synopses after this filtering process has been performed are denoted by $\bar{B}$, respectively. Now, BS makes an estimate of the expected length of prefix of all '1's, r using $\bar{B}$. Let $\hat{r}$ be the estimate of r. We observe that there is one factor which could possibly deviate the estimate $\hat{r}$ from r: injection of false MACs by the adversary—which can cause BS not receiving any valid MAC for a few '1' bits near bit r in synopsis B. We observe that this factor could contribute to a deviation to the left only (i.e. making $\hat{r}$ less than r), as shown later (in Section V-D).

2) Phase Two: BS requests the nodes which contribute to bits i, $i > \hat{r}$, in the synopsis to send back the corresponding MACs. The message sent by BS is as follows:

$$BS \rightarrow\rightarrow * : \langle \text{“}PhaseTwo\text{”}, \hat{r} \rangle,$$

where “*PhaseTwo*” is a flag indicating that phase two is going to begin.

After receiving the request from BS, each node X broadcasts to its parents the MACs, $\{M_i \mid \hat{r} < i \leq \eta\}$. Unlike the first phase, now no MAC is dropped by the intermediate nodes, i.e, each node X forwards to X's parents all of the MACs X received from its child nodes.

After BS receives the MACs, any bit B_i, $i > \hat{r}$ for which a valid MAC is received is set to '1'. The resulting synopsis is denoted by B'.

3) An Example: Let us illustrate the critical points of the above protocol with the same example as depicted in Figure 2. Node P has 3 child nodes X, Y, and Z. In phase one, each node is supposed to send (to its parent) its fused synopsis along with one randomly selected MAC for each '1' bit. So, X sends 111101010000 as its fused synopsis, and a few MACs such as M_i, $1 \leq i \leq 4$, M_6, and M_8. Note that X may have received some of these MACs from its child nodes and not sure if some of them are false MACs. Likewise, Y sends 111110000000, M_i, $1 \leq i \leq 5$, and Z sends 111000100000, M_i, $1 \leq i \leq 3$, and M_7. If P is not malicious, then P should compute its fused synopsis as 111111110000 , and should randomly select one MAC for each '1' bit from the received MACs. P should forward M_i, $1 \leq i \leq 8$. However, if P is malicious, it injects false '1's, and forwards 111111111111 (as its fused synopsis), and M_i, $1 \leq i \leq 12$, where (at least) M_9, M_{10}, M_{11}, and M_{12} are false MACs. At the end of phase one, BS will be able to detect that M_9, M_{10}, M_{11}, and M_{12} are false MACs if no node sent a valid MAC for these four bits. Furthermore, let us assume that for the 8-th bit all MACs (e.g. one was sent by X) received by BS are false. Then, BS's estimate $\hat{r}$ can be 7 and it requests the network to start the next phase. In phase two, each node which generates valid MAC for bit 8, 9, 10, 11 or 12 sends such MACs, and all of these MACs should reach BS (no random selection of MACs en-route), and BS can determine the correct status of each bit of the synopsis.

D. Correctness and Security Analysis

In particular, we prove that BS can correctly infer the values of all of the bits in the synopsis. In other words, we show that when this protocol terminates, BS has already received at least one valid MAC for each '1' bit of the synopsis.

1) Correctness of the Protocol Phases:

a) **Phase One:** There exists some room for the adversary to shift the estimate, $\hat{r}$ from the true value of r. However, below we show that only the left shift is possible.

Claim V.1. *After phase one, BS's estimate of the prefix-length (of '1' bits in the synopsis) $\hat{r}$ cannot be higher than the true prefix-length r, i.e. $\hat{r} \leq r$.*

Proof: By contradiction: Say, if possible, $\hat{r} > r$. By definition of r, bit $(r+1)$ in the true synopsis is '0', i.e. there

is no node in the network, which contributes to bit $(r+1)$ and sends a valid MAC to BS. So, the only possibility is a compromised node generated a false MAC (and sent to BS), and BS was not able to detect the falsehood of this MAC. This is not possible according to Lemma V.3. Hence, this Claim is proved.

■

b) **Phase Two:** This phase addresses the above deviation (of $\hat{r}$ from r) by validating bits to the right of bit $\hat{r}$ to securely obtain the position of bit r.

Claim V.2. *After phase two ends, the finally computed synopsis B' at BS is correct, i.e. the prefix-length of B' is same as the true prefix-length r.*

Proof: In phase two, nodes do not employ a random selection procedure in forwarding MACs; each node forwards to its parent nodes all of the received MACs. We assume that (due to the use of multi-path routing) at least one MAC for each bit will reach BS. So, combining the contribution of phase one and phase two, BS is able to correctly determine all of the bits $B[i]$, $i \geq 1$ for the synopsis.

■

Note that strictly speaking, to get the above results, we make assumptions that there exists at least one communication path from each node to BS and it is true during both the phases. We stress that there can be an isolated node, but then its value should not be considered in the aggregate (e.g. Sum)'s definition; so, this scenario does not pose any challenge to the correctness of our protocol. Below we observe that the above assumption holds in a practical network. Let α be the packet loss rate and each node has at least f parents in the aggregation hierarchy. Then, the probability of node X's MAC to reach BS is greater than $(1-\alpha^f)^g$, where node X is g hops away from BS. As an example, if $\alpha = 0.1$, $f = 3$, and $g = 5$, then this probability is more than 99.5%. Furthermore, a compromised node C's dropping a MAC generated by node X which contributes a '1' to bit B_i has no effect if there is another node Y which also contributes a '1' to bit B_i and hence sends its own MAC. To stop all of the MACs corresponding to bit B_i from reaching BS, the attacker has to compromise all of the possible paths from all of these contributing nodes to BS, which is hard to achieve. We note that there are bits in the synopses to which only one or two nodes contribute. However, it is very hard for the attacker to predict in advance which nodes will be contributing to these particular bits. As a result, our protocol correctly works even in the presence of the attack.

2) Security Analysis: We first present an important observation as Lemma V.3, which acts as the basis of our analysis.

Lemma V.3. *The adversary cannot generate a MAC associated to a false '1' bit in $\hat{B}$ which BS will not be able to detect as false. (We assume that the brute-force MAC generation attack does not succeed.)*

Proof: Recall from Section V-A that if node X contributes to bits $b_1, b_2, \ldots, b_\zeta$ in its local synopsis Q^X, it generates a MAC, $M = MAC(K_X, L)$, where K_X is the key that node X shares with BS and the format of L is $< X, v_X, b_1, b_2, \ldots, b_\zeta, Seed >$. Each node X appends L' with M where $L' =< X, v_X, b_1, b_2, \ldots, b_\zeta >$.

Let us consider that a compromised node X's MAC, M reaches BS. First, we observe that use of MACs ensures that node X cannot inject a MAC on behalf of another node without being detected. We also observe that M cannot vouch for a false '1' at bit i because of the following reason. To vouch for a false '1' at bit i, i has to be appended in the bit list in L. As a result, BS will detect its falsity after re-executing the Synopsis Generation Algorithm (Algorithm 2) with parameters as X and the sensed value, v_X. Note that using the same *Seed* ensures that in the above process BS generates exactly same synopsis as Q^X. So, the only option for X to successfully inject a false '1' is to modify v_X (i.e. launching the falsified local value attack). ■

Lemma V.3 implies that a compromised node X cannot successfully inject a false yet undetected '1' bit via falsified sub-aggregate attack—to inject such a '1', X has to launch falsified local value attack. In fact, the maximum error the compromised node X can inject (without being detected) to the final aggregate is $(v_{max} - v_X)$, where v_X is X's actual local value and v_{max} the maximum possible local value. Refer the Appendix for more details.

We assume that no compromised node is interested in sending false MACs in phase two. Note that if a compromised node does so, this attack would have no impact in corrupting the aggregate at BS. This attack can only jam the network with bogus packets, i.e. can effectively create a denial-of-service situation, which is out of scope of this paper.

E. Performance Analysis

The communication overhead of phase one does not depend on the number of compromised nodes. The worst case per-node communication burden is to forward l MACs, where l is the maximum number of '1's in the synopsis. From property 1 of Sum synopsis (Section III-B), we know that l is approximately $\log_2 S$, S being the Sum. That means the communication overhead per node is $O(\log_2 S)$.

On the other hand, the communication overhead of phase two is determined by how close the estimate $\hat{r}$, obtained in phase one, is to the real value of r.

First, we present Lemma V.4 and Lemma V.5, which we will use to obtain the subsequent results.

Lemma V.4. *Let a synopsis be computed to determine the Sum using the FM algorithm [15], where S is the value of the Sum. Let r be the expected value of R in the synopsis B at BS. The probability that $B[i] = 0$, with $r - a \leq i \leq r + a$ and $a \geq 0$, is:*

$$Pr[B[i] = 0] \approx e^{-\frac{(1/\phi)}{(2^{i-r})}},$$

where $\phi = 0.7735$.

Proof:

From Property 2 of Sum synopsis (ref. Section III-B), we see that r is $\log_2(\phi S)$. As observed in Section III-B, the function $CT()$ in SG_{sum} algorithm is invoked S times in total considering synopsis generation of all nodes. A bit j in the

final synopsis B is '0' only if none of the above S invocations of $CT()$ returns j. So, we see that:

$$Pr[B[j]=0]=(1-\frac{1}{2^j})^S.$$

To calculate the probability of r-th bit[2] being '0', we substitute j with $r=\log_2(\phi S)$ and we get:

$$Pr[B[r]=0]=(1-\frac{1}{2^r})^S=(1-\frac{1}{\phi S})^S\approx e^{-\frac{1}{\phi}}.$$

In general, for the i-th bit where $r-a\le i\le r+a, a\ge 0$, we get:

$$Pr[B[i]=0]=(1-\frac{1}{2^i})^S=(1-\frac{1}{2^{i-r}2^r})^S=(1-\frac{1}{2^{i-r}\phi S})^S\approx e^{-\frac{(1/\phi)}{(2^{i-r})}}.$$

■

We observe from Lemma V.4: The probability that $B[i]=0$ is determined by only the distance of the i-th bit from the r-th bit, where the value of r is $\log_2(\phi S)$. Furthermore, in Lemma V.5 below, we observe that the bits close (left or right) to bit r or far to the right of bit r can be considered as independent.

Lemma V.5. *Let r be the expected value of R in the synopsis B at BS. Let the value of Sum be S. The value ('0' or '1') of any two bits B_i and $B_{i'}$, with $i\ge(r-b)$, $i'\ge(r-b)$, $i\ne i'$, $b << log_2 S$ are independent.*

Proof: By construction of SG_{sum} algorithm (ref. Section III-B), we get the following two relations:

$$Pr[B[i]=0]=(1-\frac{1}{2^i})^S \tag{2}$$

$$Pr[B[i]=0,B[i']=0]=(1-\frac{1}{2^i}-\frac{1}{2^{i'}})^S \tag{3}$$

Then, by basic rules of probability, we have:

$$Pr[B[i]=0\mid B[i']=0]=\frac{Pr[B[i]=0,B[i']=0]}{Pr[B[i']=0]}=\frac{(1-\frac{1}{2^i}-\frac{1}{2^{i'}})^S}{(1-\frac{1}{2^{i'}})^S} \tag{4}$$

From Relation 2 and Relation 4, we get:

$$\frac{Pr[B[i]=0|B[i']=0]}{Pr[B[i]=0]}=\frac{(1-\frac{1}{2^i}-\frac{1}{2^{i'}})^S}{(1-\frac{1}{2^{i'}})^S(1-\frac{1}{2^i})^S}=(1-\frac{1}{(2^i-1)(2^{i'}-1)})^S \tag{5}$$

So, we have:

$$\frac{Pr[B[i]=0|B[i']=0]}{Pr[B[i]=0]}\approx(1-\frac{1}{2^i2^{i'}})^S\approx e^{-\frac{S}{2^i2^{i'}}}\le e^{-\frac{S}{(\frac{\phi S}{2^b})(\frac{\phi S}{2^b})}}=e^{-\frac{2^{2b}}{\phi^2 S}}\approx 1 \tag{6}$$

■

We remind the reader that a practical attack can only move the estimate $\hat{r}$ to the left. We present Claim V.6 to quantify the deviation.

Claim V.6. *Let $\hat{r}$ be the estimate of the position of bit r after phase one, and let t denote the number of compromised nodes in the network. There is a δ_a such that $Pr[(r-\hat{r})>\delta_a]\approx 0$.*

[2]Strictly speaking, as r is a real number, 'r-th bit' is not a well-defined bit. However, as we are making an average case analysis, we refer to 'bit r' for ease of exposition.

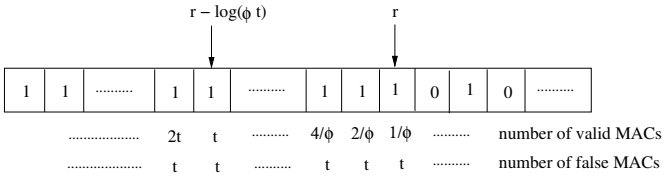

Fig. 3. The Synopsis at BS and a High-level View of the Random Selection Procedure in MACs Forwarding—The number of contributing nodes and hence the number of valid MACs varies with the bit position of the synopsis, e.g. for the r-th bit this number is $\frac{1}{\phi}$, and so on. The maximum number of false MACs is t for any bit when t nodes attack.

Proof: We now consider the attack in which compromised nodes inject false MACs in phase one. Recall that we enforce a restriction on the number of MACs that a node can inject into the network. In fact, if node X sends an aggregation message (synopsis $\hat{B^X}$ and corresponding MACs) to node Y, Y does not accept more than one MAC for each '1' bit in $\hat{B^X}$.

Without loss of generality, we focus on one particular '1' bit, say B_i i.e., the i-th bit in synopsis B. Let us assume that there are s nodes in the network which contribute to this bit and hence each of these s nodes sends a MAC toward BS. On the other hand, if there are t compromised nodes present in the network, then there can be at most t injected false MACs for this bit. We can consider that these s valid MACs and t false MACs compete with one another in the random selection procedure at the intermediate hops, and finally, only one MAC reaches BS. We assume that these s good nodes and t bad nodes are randomly distributed in the network. So, we can consider that a valid MAC finally reaches BS with probability $s/(s+t)$.

Recall that the number of contributing nodes varies exponentially with the bit position—for the r-th bit, the expected number of contributing nodes is $1/\phi$, for the $(r-1)$-th bit $2/\phi$, and so on. Let $r_a=r-\log(\phi t)$. For bit r_a-j, the expected number of contributing nodes is $2^j t$ (as illustrated in Figure 3). Hence, the probability that a valid MAC reaches BS for this bit is $p_{(r_a-j)}=(2^j t)/(2^j t+t)=(2^j)/(2^j+1)$. We observe that this probability is approximately 1 while $j>4$. Hence, this claim is proved.

■

As a special case, we see that for $\delta_a=\log\phi t+1$, $Pr[\hat{r}<(r-\delta_a)]<0.01$. That is, even during the attack the estimate $\hat{r}$ of phase one is no less than $r-\log_2\phi t-1$ with high probability. It implies that by Property 4 of Sum synopsis (ref. Section III-B) at most $O(t)$ MACs are expected to be transmitted in total over the network in phase two.

F. Comparing with Existing Approaches

As discussed in Section II-B, the attestation phase of SDAP algorithm [10] can be used to filter out the false sub-aggregates injected by the compromised nodes. In addition, a sampling-based attack resilient algorithm has been recently proposed in [11]. Moreover, our previous paper presented an attack-resilient aggregation algorithm [13] in the synopsis diffusion framework. To the best of our knowledge, these are the only works present in the literature which address the problem

of computing aggregates despite the adversarial interference. Table IV presents a comparative study of our attack-resilient computation protocol (Section V-C), our prior work [13], SDAP [10], and sampling-based approach [11].

TABLE IV
COMPARING THE ATTACK-RESILIENT PROTOCOLS

Protocols	Latency	Communication overhead	Number of keys stored in each node
Our current protocol	2 epochs	$max(O(m\log S), O(mt))$	$O(1)$
[13]	$O(\frac{\log\chi}{w''})$ epochs (w.h.p.)	$O(m\cdot 2^{w''})$	$O(1)$
SDAP by [10]	2 epochs	$O(N)$ (worst case)	$O(1)$
Sampling -based [11]	$O(\log N)$ epochs	$O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\log S)$	$O(\log S)$

Latency: Our current attack-resilient protocol takes 2 epochs [3], and with the attestation phase included, SDAP [10] takes the same number of epochs. The worst case latency incurred in [13] is $O(\frac{\log\chi}{w''})$ in most of the cases, where χ is the ratio of the upper bound of Sum to the actual value and w'' is the size of the sliding window used. Note that if the upper bound of Sum is loose, then χ can be large and in that case [13] can incur high latency. The sampling-based protocol [11] takes $O(\log N)$ epochs to complete, where N is the network size.

Communication Overhead: In SDAP, the communication overhead of the attestation phase depends on the topology of the network; for an irregular network, the worst case node congestion is O(N). In our current protocol, a node needs to forward $max(O(\log S), O(t))$ MACs in the worst case for each synopsis to compute Sum, S. To be fair to other protocols, here we report the total communication overhead which is incurred in computing all the synopses: a node may need to forward $max(O(m\log S), O(mt))$ MACs where m is the number of synopses used. The worst case node congestion in [13] is $O(m\cdot 2^{w''})$ where w'' is the sliding window size. The per-node communication overhead in [11] is $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\log S)$ to produce an (ε,δ)-approximate estimate of Sum, S. To produce a similarly accurate estimate of Sum, the value m in our algorithms should be $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ as analyzed in [9].

Number of Stored Keys: For [11], each node has to store $O(logS)$ symmetric keys which are shared with the base station. On the other hand, for other protocols, each node stores $O(1)$ keys.

Robustness to Message Loss: Communication loss may disrupt SDAP [10] algorithm. In contrast, our current protocol and that in [13] or [11] are robust against loss because they use multi-path routing schemes.

VI. SIMULATION RESULTS

The simulation study examined the correctness and performance of our algorithm. The evaluation is done based on metrics, such as closeness of the simulation results to the theoretical predictions and communication overhead. Note that we do not need to examine the latency of the protocol as it is always two epochs.

[3]As defined in the prior work [5], an epoch represents the amount of time a message takes to reach BS from the farthest node on the aggregation hierarchy.

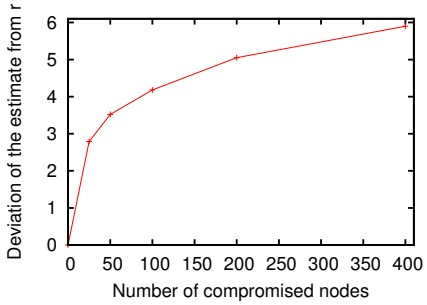

Fig. 4. Impact of Compromised Nodes on the Deviation of Phase One's Estimate, $\hat{r}$ from r (true prefix length of '1's)

A. Simulation Framework

We used the TAG simulator [5] as the basic platform on which our simulations were written. We reused some of the modules provided by Considine et al. [7], which simulate their multi-path aggregation algorithms in the TAG simulator environment. Moreover, we had to extensively expand most of their modules and add a few new modules. In particular, we had to enable two-phase-wide communications, and we had to add the security functionality (i.e. attack resilience). As the basic network topology, we used a 50×50 grid with 2500 sensor nodes, where one sensor is placed at each grid point and BS is at the center of the grid, as in [7]. The communication radius of each node is $\sqrt{2}$ unit, allowing the nearest eight grid neighbors to be reached.

We assigned a unique ID to each sensor, and each sensor reading was a random integer uniformly distributed in the range of 0 to 100 units. BS sets the synopsis (bit vector) total length as 24 bits. We used the method of independent replications as our simulation methodology. If not mentioned otherwise, each simulation experiment was repeated 300 times with a different seed. We computed the 99% confidence intervals; unless reported explicitly, the confidence intervals are within $\pm 2\%$ of the reported value.

Some of the experiments, as discussed later, consider that a few (say t) nodes among the node population are compromised, and the compromised nodes are randomly distributed in the grid. In each run of the experiment, the compromised nodes are randomly selected, but they do not change across the two phases of any run. In phase one, each compromised node invokes false MAC injection attack to the worst degree, i.e. it forwards a false MAC for each bit position (of the synopsis) to its parent nodes. To study the worst case performance of our algorithm, we consider that the compromised nodes behave as normal nodes in phase two.

B. Results and Discussion

Error in estimate $\hat{r}$. We recall from Section V that the performance of this protocol primarily depends on the looseness

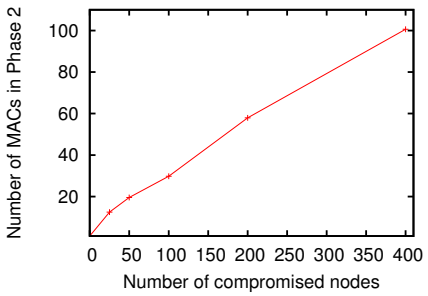

Fig. 5. Total Number of (Unique) MACs Forwarded in Phase Two.

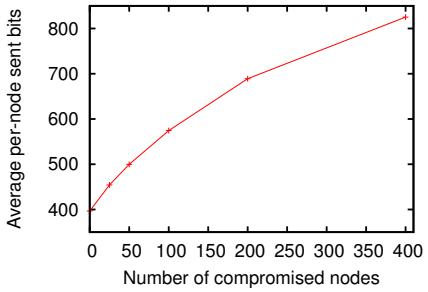

Fig. 6. Impact of Compromised Nodes on the Communication Overhead.

of the estimate, $\hat{r}$ obtained in phase one. Furthermore, the maximum deviation in estimate $\hat{r}$ from correct r (which is obtained in phase two) depends on how many compromised nodes participate in the false MAC injection attack during phase one. The analysis in Section V-E states that the deviation obeys the following inequality with high probability, $(r-\hat{r}) \leq \log_2 \phi t + 1$ where t is the number of compromised nodes. For any particular value of t (0, 25, 50, 100, 200, and 400), we simulated the false MAC injection attack during phase one 300 times. We measured $(r-\hat{r})$ for each t, and we observed that $(r-\hat{r})$ was low as expected. Figure 4 illustrates how this deviation $(r-\hat{r})$ varies with t. The 99% confidence intervals are within $\pm 10\%$ of the reported value. Moreover, in separate experiments (not shown in the figure), we also verified that the amount of the deviation does not significantly change with different network sizes (varied across 900, 1600, 2500, and 3600) if t remains same (at 50).

Communication overhead. We recall that during phase one each node needs to forward at most η MACs, where η is the length of the synopsis. The communication overhead on a node in phase two depends upon how many nodes contribute to bits to the right of bit $\hat{r}$ in the synopses because each of these nodes send a MAC to the BS. Our analysis in Section V-E shows the total number of MACs sent in the network during phase two is likely to be $O(t)$. Figure 5 plots the number of MACs sent during phase two as a function of the number of compromised nodes, t. The 99% confidence intervals are within $\pm 20\%$ of the reported value. We observe that the number of MACs increases linearly with t, which confirms our analysis. To report the total (over phase one and phase two) per-node communication overhead, we include Figure 6 which illustrates how this (in bits) varies with t. We assume that each MAC is of size 32 bits and it is sent along with the generator node Id of length 16 bits. Furthermore, in separate experiments (not shown in the figure), we observed that the per-node communication overhead does not increase with the network size (varied across 900, 1600, 2500, and 3600), which shows that our algorithm is scalable.

VII. Conclusion

We discussed the security issues of in-network aggregation algorithms to compute aggregates such as predicate Count and Sum. In particular, we showed the falsified sub-aggregate attack launched by a few compromised nodes can inject arbitrary amount of error in the base station's estimate of the aggregate. We presented an attack-resilient computation algorithm which would guarantee the successful computation of the aggregate even in the presence of the attack.

Appendix

One can be interested to know the degree of damage inflicted by the falsified local value attack. Below we discuss this.

Lemma VII.1. *The maximum error the compromised node X can inject (without being detected) to the final aggregate is $(v_{max} - v_X)$, where v_X is X's actual local value and v_{max} the maximum possible local value.*

Proof: We notice that in Algorithm 2, a node X runs the while loop v_X times. Over these runs, once a bit is set to '1', it is never reset. This implies that only by increasing the value of v_X, node X gets a chance to inject a false '1'. Recall that injecting '1's in the synopsis results in an increase in the value of the final aggregate. So, X can contribute the maximum error to the final aggregate if X generates its local synopsis Q^X considering $v_X = v_{max}$. ■

Lemma VII.1 implies that the maximum total error t compromised nodes can inject (without being detected) to the final aggregate is at most $t(v_{max} - v_{min})$, where v_{min} is the minimum sensed value.

References

[1] Mingyan Liu, Neal Patwari, and Andreas Terzis. Scanning the issue. *Proceedings of the IEEE: Special issue on sensor network applications.*, 98(11):1804–1807, 2010.

[2] Teresa Ko, Josh Hyman, Eric Graham, Mark Hansen, Stefano Soatto, and Deborah Estrin. Embedded imagers: Detecting, localizing, and recognizing objects and events in natural habitats. *Proceedings of the IEEE: Special issue on sensor network applications.*, 98(11):1934–1946, 2010.

[3] Peter Corke, Tim Wark, Raja Jurdak, Wen Hu, Philip Valencia, and Darren Moore. Environmental wireless sensor networks. *Proceedings of the IEEE: Special issue on sensor network applications.*, 98(11):1903–1917, 2010.

[4] James Reserve Microclimate and Video Remote Sensing. http://research.cens.ucla.edu/projects/2006/terrestrial/microclimate/default.htm, 2006.

[5] S. Madden, M. J. Franklin, J.M. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad hoc sensor networks. In *Proc. of 5th USENIX Symposium on Operating Systems Design and Implementation*, 2002.

[6] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring sensor networks. In *Proc. of the 2nd Int'l Workshop on Sensor Network Protocols and Applications*, 2003.

[7] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proc. of IEEE Int'l Conf. on Data Engineering (ICDE)*, 2004.

[8] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proc. of the 2nd international conference on Embedded networked sensor systems (SenSys)*, 2004.

[9] M. Garofalakis, J. M. Hellerstein, and P. Maniatis. Proof sketches: Verifiable in-network aggregation. In *Proc. of the 23rd Int'l Conference on Data Engineering (ICDE)*, 2007.

[10] Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. In *Proc. of ACM MOBIHOC*, 2006.

[11] Haifeng Yu. Secure and highly-available aggregation queries in large-scale sensor networks via set sampling. In *Proc. of the Int'l Conference on Information Processing in Sensor Networks*, 2009.

[12] Sankardas Roy, Mauro Conti, Sanjeev Setia, and Sushil Jajodia. Secure data aggregation in wireless sensor networks. *IEEE Transactions on Information Forensics and Security*, 7(3):1040–1052, 2012.

[13] Sankardas Roy, Sanjeev Setia, and Sushil Jajodia. Attack-resilient hierarchical data aggregation in sensor networks. In *Proc. of ACM Workshop on Security of Sensor and Adhoc Networks (SASN)*, 2006.

[14] M. B. Greenwald and S. Khanna. Power-conservative computation of order-statistics over sensor networks. In *Proc. of the 23th SIGMOD Principles of Database Systems (PODS)*, 2004.

[15] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[16] L. Hu and D. Evans. Secure aggregation for wireless networks. In *Proc. of Workshop on Security and Assurance in Ad hoc Networks.*, 2003.

[17] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2006.

[18] D. Wagner. Resilient aggregation in sensor networks. In *Proc. of ACM Workshop on Security of Sensor and Adhoc Networks (SASN)*, 2004.

[19] L. Buttyan, P. Schaffer, and I. Vajda. Resilient aggregation with attack detection in sensor networks. In *Proc. of 2nd IEEE Workshop on Sensor Networks and Systems for Pervasive Computing*, 2006.

[20] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *Proc. of the 1st Int'l Conference on Embedded networked sensor systems (SenSys)*, 2003.

[21] Keith Frikken and Joseph A. Dougherty. An efficient integrity-preserving scheme for hierarchical sensor aggregation. In *Proc. of the 1st ACM Conference on Wireless Network Security (WiSec)*, 2008.

[22] Suman Nath, Haifeng Yu, and Haowen Chan. Secure outsourced aggregation via one-way chains. In *Proc. of the 35th SIGMOD international conference on Management of data*, pages 31–44, 2009.

[23] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Proc. of the International Conference on Mobile Computing and Networks (MobiCOM)*, 2001.

[24] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. F. Abdelzaher. Pda: Privacy-preserving data aggregation in wireless sensor networks. In *Proc. of IEEE IEEE International Conference on Computer Communications (INFOCOM)*, 2007.