Towards Statistically Strong Source Anonymity for Sensor Networks

Min Shao, Yi Yang, Sencun Zhu, Guohong Cao
The Pennsylvania State University
E-mail: {mshao,yy5,szhu,gcao}@cse.psu.edu

Abstract—**For sensor networks deployed to monitor and report real events, event source anonymity is an attractive and critical security property, which unfortunately is also very difficult and expensive to achieve. This is not only because adversaries may attack against sensor source privacy through traffic analysis, but also because sensor networks are very limited in resources. As such, a practical tradeoff between security and performance is desirable. In this paper, for the first time we propose the notion of *statistically strong source anonymity*, under a challenging attack model where a global attacker is able to monitor the traffic in the entire network. We propose a scheme called *FitProbRate*, which realizes statistically strong source anonymity for sensor networks. We also demonstrate the robustness of our scheme under various statistical tests that might be employed by the attacker to detect real events. Our analysis and simulation results show that our scheme, besides providing source anonymity, can significantly reduce real event reporting latency compared to two baseline schemes.**

Index Terms—**security and privacy, source anonymity, statistical test, SPRT, sensor networks**

I. INTRODUCTION

Sensor networks have been envisioned to be very useful for a broad spectrum of emerging civil and military applications [1]. However, sensor networks are also confronted with many security threats such as node compromise, routing disruption and false data injection, because they normally operate in unattended, harsh or hostile environment.

Among all these threats, privacy (especially source anonymity) is of special interest since it cannot be fully addressed by traditional security mechanisms such as encryption and authentication. Consider a simple example of event reporting in sensor networks (as shown in Fig. 1). When a sensor detects an event, it sends a message including event related information to the base station. If an attacker (the hunter here) can intercept the message, it may know such sensitive information as whether, when and where a concerned event has happened, e.g., the appearance of an endangered animal in a monitoring sensor network [2]. Following this, the attacker can take some action to capture/kill the animal.

To preserve source anonymity is a challenging task in sensor networks that have scarce resources in energy, computation and communication. Hence, only lightweight, energy efficient privacy-conserving mechanisms are affordable. Moreover, sensors typically have low-cost radio devices that employ standardized wireless communication technologies, which allow
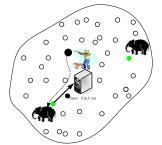
Fig. 1. An application of sensor networks for animal monitoring.

an attacker to easily monitor or eavesdrop in communications between sensors. Consequently, it is also possible for a single attacker to monitor all the network traffic either by deploying his own sensors that cover the whole deployment area or by employing a powerful site surveillance device with hearing range no less than the network radius.

Despite its importance, so far, sensor source anonymity has not received enough attention, and the existing solutions have limitations when directly applied to sensor networks. For example, in phantom routing [2], the attacker has limited coverage, comparable to that of sensors. Therefore, only a single source is under the attacker's consideration at a time and the attacker tries to trace back to the source in a hop-by-hop fashion. When the attacker becomes more powerful, e.g., has a hearing range more than three times that of the sensors, the capture likelihood is as high as 97%. In addition, a large number of anonymity techniques [3] designed for general networks are not appropriate to be used for sensor networks. This is not only because the privacy problem is different but also because these techniques are too expensive to be employed.

In this paper, we aim to provide source anonymity for sensor networks under a global observer who may monitor and analyze the traffic over the whole network. Clearly, if all the traffic in the network is real event messages, it is unlikely to achieve source anonymity under such a strong attack model. Therefore, we employ network-wide dummy messages to achieve global privacy. The basic idea is as follows. Every node in the network sends out dummy messages with intervals following a certain kind of distribution, e.g., constant or probabilistic. When a node detects a real event, it transmits the real event messages with intervals following the same distribution. As such, neither can an attacker discern the occurrence of a real event, nor can he find out the location of the real event source.

To reduce the extra overhead caused by dummy messages,

the message transmission rate should be relatively low. In this case, however, the real event report latency could be high, because a source node needs to postpone the transmission of a real event message to the next interval. Therefore, the questions we try to answer are: *how to achieve global source anonymity without causing high real event notification latency? Is it possible to provide perfect global privacy without losing performance benefit?*

More specifically, we make the following contributions in this paper. First, we demonstrate that it is difficult to achieve perfect global privacy without sacrificing performance benefit. Hence, we have to relax the perfect source anonymity requirement and for the first time propose a notion of *statistically strong source anonymity* for sensor networks. Second, we devise a realization scheme, called *FitProbRate (Fitted Probabilistic Rate) scheme*, in which the event notification delay is significantly reduced while keeping statistically strong source anonymity, through selecting and controlling the probabilistic distribution of message transmission intervals.

The rest of the paper is organized as follows. We first formalize the problem in Section II. After that, Section III presents the FitProbRate scheme. Its performance is evaluated in Section IV and its security property is analyzed in Section V. Finally, we describe the related work in Section VI and conclude this paper in Section VII.

II. PROBLEM FORMALIZATION

A. Network Model

As in other sensor networks [4], our system also assumes that a sensor network is divided into cells (or grids) where each pair of nodes in neighboring cells can communicate directly with each other. A cell is the minimum unit for detecting events; for example, a cell head coordinates all the actions inside a cell. Each cell has a unique id and every sensor node knows in which cell it is located through its GPS or an attack-resilient localization scheme [5]. Moreover, we assume a base station (BS) located in the network and works as the network controller to collect event data. Every event has an event id; for example, we may assign a unique id to each type of animals. When a cell detects an event, it will send a triplet (cell id, event id, timestamp), which provides the BS with the source location of the event as well as the time it is detected.

B. Adversary Model

According to the classification in [6], we assume that the adversary is *external*, *global* and *passive*. By external, we assume that the adversary does not compromise or control any sensors. By global, we assume that the adversary can monitor, eavesdrop and analyze *all* the communications in the network. The adversary may launch active attacks by channel jamming or other denial-of-service attacks. However, since these attacks are not related to source anonymity, we do not address them.

Next, we discuss how an adversary may analyze the collected traffic. First, the attacker may simply examine the content of an event message that may contain the source location id. Second, even if the message is encrypted, it is easy for the global adversary to trace back to the source of

the message if the encrypted message remains the same during its forwarding, because the adversary is capable of identifying the immediate source of a message. Third, he may perform more advanced traffic analysis including rate monitoring and time correlation. In a rate monitoring attack, the adversary pays more attention to the nodes with different (especially higher) transmission rates. In a time correlation attack, he may observe the correlation in transmission time between a node and its neighbor, attempting to deduce a forwarding path. We assume that the attacker has sufficient resources (e.g., in storage, computation and communication) to perform these advanced attacks.

C. Problem Definition

According to [7], a mechanism to achieve *anonymity* appropriately combined with dummy traffic yields *unobservability*, which is the state that items of interest (IOIs) are indistinguishable from any IOI of the same type. All the subjects under consideration constitute an *unobservability set*. In our case, the unobservability set consists of all the N cells in the network. Specifically, we are interested in *event unobservability*, which is defined as follows.

Definition 1: Event unobservability is a privacy property that can be satisfied if an attacker cannot determine whether real events have occurred through observation.

Straightforward solutions exist to provide event unobservability by means of dummy traffic. For example, in a *ConstRate* scheme, all the cells in the network send out messages at a constant rate no matter there are real events or not (messages are always encrypted by a secret key shared between a node and the base station [8]). Since the traffic in the network always keeps the same pattern, it effectively defeats any traffic analysis techniques. Clearly, the average transmission latency in a source cell is half of the interval. Although this deterministic solution provides the property of perfect event unobservability, it has an inherent difficulty in setting the constant rate. If the rate is too small, the message delay will be too high; if the rate is too large, this approach will introduce too much dummy traffic into the network.

This motivated us to design probabilistic solutions, which provide the chance of reducing the waiting time. For example, we may adopt an exponential distribution to determine the time intervals between message transmissions, which is referred to as the *ProbRate* scheme here. Under our attack model, a global attacker can easily know the distribution and its mean by a statistic test over the collected time intervals. However, when we keep the seed for generating random numbers secret from an attacker, the attacker may not be able to notice if a message transmission is due to a real event or a dummy message even if a cell sends out a real event message immediately. Intuitively, a cell cannot always send real event messages immediately in the presence of burst events; otherwise, an attacker may notice the change of the underlying distribution. Therefore, it is difficult to guarantee perfect event unobservability while providing low latency.

Hence, the question becomes: *how to reduce the latency as much as possible while still providing a satisfactory degree*

of event unobservability? In order to achieve low latency, we need to relax the perfect event unobservability requirement and accept *statistically strong event unobservability*.

Let the inter-message delay (imd) between message $k(k > 0)$ and $k+1$ from cell $i(1 \leq i \leq N)$ be $imd_k^i = t_{k+1}^i - t_k^i$, where t_k^i is the transmission time of message k from cell i. A global observer can see a sequence of continuous inter-message delays, which can be represented by a distribution $X^i = \{imd_1^i, imd_2^i, \cdots\}$. Ideally, in a scheme with perfect privacy, inter-message delays from all the cells follow the same distribution. In our case with statistically strong guarantee, distributions of inter-message delays are actually *statistically indistinguishable* from each other. Next, we first introduce the definition of statistically indistinguishable distributions.

Definition 2: Two probabilistic distributions X^i and $X^j (1 \leq i, j \leq N, i \neq j)$ are statistically indistinguishable from each other iff they follow the same type of probabilistic distribution with the same parameter (i.e., they have the same distribution function) statistically. They are indistinguishable from each other in the sense that by a statistic test one cannot differentiate them.

Take the exponential distribution as an example. This distribution has only one parameter $\lambda(= 1/\mu)$. Hence, if two probabilistic distributions are both exponential distributions with very close means, they are statistically indistinguishable from each other. Note that if a distribution is controlled by multiple parameters (e.g., two in a normal distribution), two data sets are statistically indistinguishable only when all these parameters of the probabilistic distribution derived from the two data sets are the same or very close. Clearly, the more parameters a distribution has, the harder it is to prove its statistical indistinguishability. As such, in the following we will limit our discussion on one-parameter distribution.

For the one-parameter distribution, the property of statistically strong event unobservability is related to two security parameters α and ϵ, where α controls the goodness of fit to a specific probabilistic distribution and ϵ controls the closeness of the parameter derived from the observations to that of the population. These two security parameters are used together so that message transmission time intervals from all the cells in the network, including the real sources if any, follow the "same" distribution with the "same" parameters. Here "same" means that an attacker cannot tell the difference through a statistical hypothesis test. More formally, we call this statistically strong event unobservability as (α, ϵ)-*unobservability*, because two parameters α and ϵ tightly relate to this privacy property.

Definition 3: (α, ϵ)-unobservability ($\alpha, \epsilon > 0$) is a type of statistically strong event unobservability, in which a distribution X^i (with parameter λ^i) is statistically indistinguishable from a probabilistic distribution X (e.g., exponential with parameter λ) under the following conditions:

1) $n \int_{-\infty}^{+\infty} [F(X^i) - F(X)]^2 \Psi[F(X)] dF(X) \leq c$,
2) $(1 - \epsilon)\lambda \leq \lambda^i \leq (1 + \epsilon)\lambda$,

where n is the sample size, F is a cumulative distribution function (CDF), Ψ is a weight function, and c is a critical value determined by α.

The left side of Condition 1 calculates the distance between two CDFs. Details of evaluating the distance between two distributions could be found in [9], [10]. If the distance between two distributions is smaller than a critical value determined by significance level α and their parameters are close to each other in a way determined by ϵ in Condition 2, these two distributions satisfy (α, ϵ)-unobservability. The above distance evaluation of CDFs was used in Anderson-Darling test [11] for goodness of fit tests; therefore, to achieve (α, ϵ)-unobservability our schemes will directly use Anderson-Darling tests. The above definition is rather general, which leaves a large room for defining α and ϵ according to different applications or extending it to the multiple-parameter case.

III. THE FITPROBRATE SCHEME

In this section, we discuss the building blocks of our scheme, including the policy for dummy traffic generation and the policy for embedding real event messages. Finally, a running example is used to illustrate the entire process of our scheme.

A. Policy for Dummy Traffic Generation

The transmission rate of dummy messages determines the network transmission overhead. As discussed in Section II-C, high rate causes high message overhead, whereas low rate increases the delay of reporting real events. In addition, the *ProbRate* scheme where message transmission rate follows a probabilistic distribution provides an opportunity for reducing latency, compared with the *ConstRate* scheme where message transmission rate is fixed. Hence, we prefer probabilistic message transmission intervals.

Now we need to decide what probabilistic distribution to use. There are many probability distributions; e.g., exponential, uniform, weibull, normal. The advantage of an exponential distribution is that it has only one parameter ($\lambda = 1/\mu$, where μ is the mean), which makes it relatively easy to achieve (α, ϵ)-unobservability. Therefore, to maximize the communication randomness and to simplify the problem, we choose the exponential distribution to control the rate of dummy traffic generation.

Specifically, Algorithm 1 implements our idea of probabilistic dummy traffic generation. Suppose there are a series of k dummy messages, our goal is to make the time intervals between two consecutive messages ($imd_i, i = 1, 2, \cdots, k-1$) follow an exponential distribution. Given a mean μ and a global variable $seed$, the algorithm returns the time interval to transmit the next dummy message. The mean μ of the exponential distribution is a system parameter and we assume it is known to the adversary because he can calculate it from observed message intervals. However, the seed for generating random numbers is kept secret from the adversary, and the seed is hard to guess and different for each sensor node.

B. Policy for Embedding Real Traffic

When a real event happens, by exactly following the ProbRate scheme, i.e., determining the waiting time based on Algorithm 1, in the long run, we cannot gain anything over the ConstRate scheme if the message transmission rates in

these two schemes have the same mean. On the other hand, if the real event message is sent out right away, the distribution of time intervals could be skewed (i.e., the mean becomes smaller), leaving the real event evident. So, our goal is *to keep the message transmission intervals following the same distribution while reducing the real event report latency.*

More formally, when a real event E_k happens after the dummy events $E_i(1 \le i \le k-1)$, the corresponding message should be sent out only when the next time interval (imd_k) and the earlier ones ($imd_i, i = 1, 2, \cdots, k-1$) satisfy the following two conditions:

- The whole series $\{imd_1, imd_2, \cdots, imd_{k-1}, imd_k\}$ still follow the same exponential distribution;
- imd_k is as small as possible.

From the attacker's perspective, in order to detect real event messages, he may perform a statistic test to determine if the message transmission intervals always follow the same exponential distribution of the same mean μ, after monitoring the network traffic and collecting sufficient message transmission intervals. More specifically, the statistic test can be broken into two parts: test if the distribution is exponential and test if the mean is μ. To defend against the attacker's strategies, we adopt the following techniques:

1) A statistic test called Anderson-Darling Test is adopted to keep the message intervals of each cell following an exponential distribution, controlled by parameter α;
2) A method is used to ensure the measured sample means of the distribution do not deviate too far from the true mean μ, controlled by parameter ϵ.

Next, we introduce these two techniques separately.

1) Anderson-Darling Test: Anderson-Darling Test [12] (A-D test in short) is a goodness fit test to determine if a series of data follow a certain probabilistic distribution. The basic idea is to evaluate the distance between the distribution of the sample data and a specified probabilistic distribution. If the distance is statistically significant, the data do not follow this distribution. More formally, the test is defined as follows.

- H_0: The data follows a specified distribution;
- H_a: The data do not follow a specified distribution;
- Test Statistic: $A^2 = -n - S$, where

$$S = \sum_{i=1}^{N} \frac{2i-1}{n}[\log F(X_i) + \log(1 - F(X_{n+1-i}))].$$

  Here F is the CDF of interest, n is the sample size, and X_i denotes the ith datum;

- Significance Level: α (typically equals to 0.05);
- Critical Region: The critical values for the A-D test depend on the specific distribution being tested. Tabulated values and formulas have been published by Stephens [12]. If the test statistic A^2 is greater than the corresponding critical value c, the hypothesis that the distribution is of a specific form will be rejected.

Algorithm 2 shows some details of this A-D test for an exponential distribution. The input is a series of x_i, i.e., the time interval between two consecutive messages sent out from a cell, and the output is a decision if the series follow an exponential distribution. This algorithm mainly involves a sorting and a statistic calculating operation. The time complexity for sorting is $O(n \log n)$ (e.g., by quicksort) and the time complexity for calculating the test statistic is $O(n)$, where n is the window size (the size of the input). Therefore, the complexity of this algorithm is $O(n \log n)$.

In our problem setting, we want to use the A-D test to find an appropriate inter-message delay (imd) for transmitting the real event message, such that Algorithm 2 will return TRUE when given the whole series of time intervals $\{imd_1, imd_2, \cdots, imd_{k-1}, imd\}$. Because the A-D test is a statistical test, the solution to pass the test is not unique. Therefore, the A-D test is repeated until the test is passed. Because a small but random delay is preferred, the search process for imd starts from 0, and increases in a small random pace whenever it fails the A-D test.

Algorithm 3 shows the details of the search algorithm. It has a series of time intervals as input and returns the first imd that can pass the A-D test. The selection of the granularity (INCREASEPIECE) affects the running time. We set INCREASEPIECE to be a random number between 0 and the first quartile of the input data set, as shown in the algorithm (line 2). Based on our experiments, this can achieve a relatively small delay within a relatively short time. From line 4 to line 11, the test repeats until it finds a value which can pass the A-D test or a value which cannot be accepted because the delay becomes larger than a specified upper bound (line 5), e.g., the maximum value of $imd_1, imd_2, \ldots, imd_n$. In the latter case, another $INCREASEPIECE$ will be selected (line 6) and the searching process starts over from the value of 0. The whole algorithm terminates when a proper delay is found. Because many values can pass the A-D test with the same

input, an appropriate value can be found quickly. This has been verified by experiments. With sample sizes of 20, 40, 80, 160, 320, 640, 1280, 2560, 5120 and 10240, Algorithm 3 always terminates within 2∼10 rounds.

2) Sample Mean Recovery: If there are multiple continuous real events happening, Algorithm 3 will be called repeatedly. In this case, the sample mean will gradually decrease as smaller delays are favored in Algorithm 3. According to the Central Limit Theorem, the sample means follow a normal distribution. From the perspective of an attacker, every time he observes a new time interval, he will need to make a "yes" or "no" decision on whether a real event has occurred. If "yes", he will take an action (e.g., to check the suspicious cell by himself); otherwise, he will do nothing. However, when he makes a "yes" decision, it is possible that it is a wrong decision. Thus, as a balance between false positive rate and false negative rate, an attacker needs to determine a threshold. Once the difference between the sample mean and the true mean is beyond this threshold, he will consider the occurrence of a real event and take an action.

Thus, we need to deliberately recover the sample mean so that it will never deviate from the true mean beyond this threshold. Specifically, in our scheme we will set this threshold as $\epsilon\mu$, because in Definition 3 the condition $(1-\epsilon)\lambda \leq \lambda^i \leq (1+\epsilon)\lambda$ is equivalent to $(1-\epsilon)\mu \leq \mu^i \leq (1+\epsilon)\mu$ for the exponential distribution with $\lambda = 1/\mu$. We will search for an appropriate new time interval for the next message (real or dummy event) such that the sample mean of the entire time series including the new one is within $\epsilon\mu$ from the true mean. Algorithm 4 serves this purpose. It calculates the value needed to recover the mean (line 3) and a random number is selected between this value and a value following exponential distribution with mean μ (line 4) until this random number can pass A-D test (line $5-8$).

From the above discussions, the significance level α defined in the A-D test is used to control the acceptable distance between the observed distribution of message transmission time intervals and the exponential distribution. ϵ reflects an acceptable difference between the sample mean and the true mean, which will not cause suspicion from the attacker. With these two parameters, our FitProbRate scheme can achieve the statistically strong source anonymity defined by (α, ϵ)-unobservability.

C. A Running Example

To illustrate the whole process, a running example is shown in Fig. 2. According to Algorithm 1, three dummy messages are supposed to be sent out at time A, B and E, respectively. At time C, a real event happens, so Algorithm 3 is called and this real event is sent out at time D. After this, according to Algorithm 4, the dummy message at time E is canceled and rescheduled at time F. From the attacker's point of view, he can only see the intervals between A and B, B and D, D and F, which follow an exponential distribution and the mean is stable. Thus, the attacker cannot tell if any real event has happened.

Fig. 2. A running example to illustrate the entire process.

All algorithms can be easily implemented in sensor networks because they only involve simple operations. For example, TinyOS supports all functions used in our algorithms such as log and exp. These algorithms can be further optimized. For example, in Algorithm 2, the calculation of S involves a summation of n values. Whenever Algorithm 3 calls the A-D test (Algorithm 2), $n-1$ values in the time series are the same as that in the previous call. Thus, only one additional log and one additional exp operations are needed.

IV. PERFORMANCE EVALUATIONS

In this section, we compare the performance of the Fit-ProbRate scheme, the ConstRate scheme, and the ProbRate scheme.

A. Comparison between FitProbRate and ConstRate

In the simulation, the mean of dummy message intervals is 20s. Real events arrive according to a Poisson Arrival process with the mean changing from 1/20 to 1/100. Fig. 3 shows the delay to send a real event in both schemes. As can be seen, the average latency in the FitProbRate scheme is less than 1s, whereas the average latency in the ConstRate scheme is 10.87s, which indicates that FitProbRate can significantly reduce the transmission delay of the real event messages.
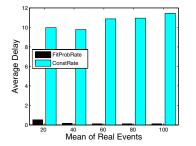

Fig. 3. Comparing average delay in the FitProbRate scheme ($\alpha = 0.05$, $\epsilon = 0.1$) with the ConstRate scheme.

B. Comparison between FitProbRate and ProbRate

In this simulation, dummy messages are generated at the average rate of 1/40s and the simulation runs for a total of 3600-second simulation time. For easy illustration, we only show part of the simulation result in Fig. 4. In the ProbRate scheme, real event messages and dummy messages are treated equally; that is, their transmission time intervals are determined by the output of Algorithm 1. To make a more comprehensive comparison, we examine three traffic patterns at different levels of burstiness for real event message generation, as shown in three different columns of Fig. 4.

In Fig. 4(a), each real event message arrives at the time point according to an exponential distribution; in Fig. 4(b) and (c), three and five real event messages are generated in a burst, at the same time points as in Fig. 4(a). Figs. 4(d)-(f) visualize the time slots at which real event messages are ready as shown by the solid lines. The dotted lines are the time points when real event messages are actually forwarded. From these figures, we can observe that real event messages are forwarded more frequently in our scheme than the ProbRate scheme. As a result, the transmission latencies of real event messages in our scheme will be much smaller than that in ProbRate.

Figs. 4(g)-(i) quantify these observations. As shown in the figure, the FitProbRate scheme can significantly reduce the real event message forwarding latency compared with the ProbRate scheme. If the real events happen in burst, the latency will be higher. As traffic pattern 3 is more bursty than traffic pattern 1, the average delay in Fig. 4(i) is also much higher than that of Fig. 4(g). This is because the average waiting time must increase to recover from mean skewness when more real messages need to be sent out within a certain time.

V. SECURITY ANALYSIS

We first prove that the FitProbRate scheme has the property of (α, ϵ)-unobservability. Then, we show the robustness of our scheme against various powerful statistical tests.

A. Security Property

We have the following theorem on the security property of the FitProbRate scheme.

Theorem 1: The FitProbRate scheme has the property of (α, ϵ)-unobservability.

Proof: To prove that the FitProbRate scheme has the property of (α, ϵ)-unobservability, we need to prove that the statistically strong event unobservability has been achieved under the control of parameters α and ϵ.

Under the control of parameter α, by Algorithm 3 the distribution X^i of message transmission intervals from any cell $i(1 \leq i \leq N)$ can pass the A-D test. This means that the difference between the empirical cumulative distribution function (CDF) from the ordered sample data and the cumulative distribution function of the corresponding exponential distribution X is smaller than the critical value c decided by the predetermined significance level α, according to the nature of A-D test. Namely, the following formula holds: $n \int_{-\infty}^{+\infty} [F(X^i) - F(X)]^2 \Psi[F(X)] dF(X) \leq c$, where n is the sample size, F is the CDF and Ψ is the weight function of the goodness of fit test.

Moreover, under the control of parameter ϵ, once the sample mean μ^i from any cell i deviates from the population mean μ of the exponential distribution in a noticeable way, i.e., $|\mu^i - \mu| \geq \epsilon$, Algorithm 4 will be automatically triggered to recover the mean. Hence, the sample mean from any cell in the network cannot be differentiated from the population mean under the control of ϵ. That is, at any time $(1 - \epsilon)\mu \leq \mu^i \leq (1 + \epsilon)\mu$.

In summary, probabilistic distributions of message transmission intervals from real sources are statistically indistinguishable from those of other cells that send out dummy messages. By Definition 3, we say the FitProbRate scheme has the property of (α, ϵ)-unobservability. ∎

Assuming the employment of our scheme, we consider what the attacker can do to detect real events. Clearly, we cannot limit an attacker from using any statistical tool, so what we show below are based on our guessing of the general while powerful techniques that might be used by the attacker. We believe other statistic testing methods will render the similar results.

We assume that the attacker has enough resources (e.g., in storage and computation) to collect and analyze message time intervals from all the cells in the network. Then, the attacker will try to identify sources with different distributions of message time intervals. To do this, the attacker can first conduct some goodness of fit tests to check whether the probabilistic distributions of message time intervals from every cell follow the exponential distribution. If the distribution from any cell cannot pass the goodness of fit test, the corresponding cell will be marked as a potential real source. Two widely used distribution test tools are adopted here: Anderson-Darling (A-D) test [9] and Kolmogorov-Simirnov (K-S) test [13]. For those distributions that can pass the goodness of fit test, the attacker then further performs the mean test. Those cells whose
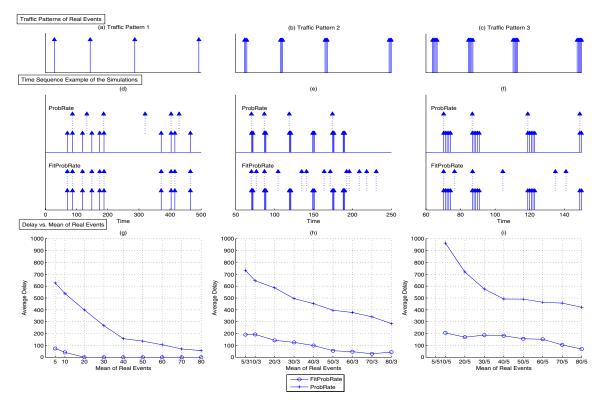
Fig. 4. Performance comparison between the FitProbRate scheme ($\alpha = 0.05$, $\epsilon = 0.1$) and the ProbRate scheme under different real traffic patterns. In (a)-(c), 1, 3, or 5 real event messages are generated in a burst. In (d)-(f) the solid lines are the time points when real events are ready and the dotted lines are the time points when real event messages are actually forwarded. (g)-(i) show the numerical values of real event transmission latency under three different real traffic patterns.

sample means deviate from the true mean beyond a certain degree will be marked as potential real sources, too. The SPRT (Sequential Probability Ratio Test) [14] is a good choice for the mean test, because SPRT could minimize the number of samples required to make a decision after continuous observations.

Next, we demonstrate the robustness of our FitProbRate scheme in defending against these testing techniques by the attacker, including its robustness to the distribution tests as well as its robustness to the mean test.

B. Robustness to Distribution Tests

To detect abnormal probabilistic distributions of message time intervals, the attacker can check whether a probabilistic distribution X^i is exponential. For the attacker, the hypotheses in the test are:

- H_0-the distribution is exponential: $F(X^i) = F(X)$.
- H_1-the distribution is not exponential: $F(X^i) \neq F(X)$.

When the attacker makes a decision, there are some risks for him to get wrong decisions. The decision is called a *detection*, if H_1 is accepted when it is actually true. If in this case H_0 is accepted, then it is called *false negative*. On the other hand, if H_0 is in fact true, accepting H_1 is a *false positive*. For the attacker, the false positive rate is denoted as α' and the false negative rate is denoted as β'. Note that in our scheme the false positive rate is actually equal to the significance level α defined in the A-D test and we denote our false negative rate as β. To differentiate from ours, the attacker's rates are denoted as α' and β' correspondingly.

One may argue that if the attacker selects a significance level α' different from that in our algorithm (α), then the attacker may detect the perturbed probabilistic distributions from the real sources. However, there is a tradeoff between false positive rate α' and false negative rate β' in attacker's distribution test. To explain this, let us consider two extreme cases. If the rejection region has critical values $-\infty$ and $+\infty$, then the attacker always accepts H_0. In this case, $\alpha' = 0$ and $\beta' = 1$. On the contrary, if the rejection region has the critical values 0 and 0, then the attacker always rejects H_0. In this case, $\alpha' = 1$ and $\beta' = 0$. Hence, it is impossible for the attacker to make both α' and β' arbitrarily small for a fixed sample size n. If the attacker chooses a very small α' in the test, then he is at the risk of having a high β', which means he has a high chance of failing to detect real events. Likewise, if the attacker chooses a smaller β', then the attacker will examine more fake sources, wasting more of his resources and time (for travelling to the fake sources).

Next we use simulations to verify the above statement. To test false positive, we generate 10,000 groups of pure exponential distributed data. Then we perform K-S and A-D test on them separately under different significance levels (ranging from 0.01 to 0.1). Finally, based on the test results, we get the false positive rate as shown in Figure. To test false negative, instead of using pure exponential distributed data, we add real event disturbance into the data and perform the same test.
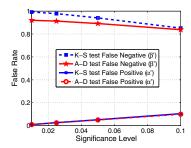
Fig. 5. A tradeoff between α' and β' for the attacker ($\alpha = 0.05$).

In Fig. 5, the x-axis is the significance level used by the attacker and the y-axis represents the false rate (either false positive or false negative). We can observe that the false negative rate β' of the attacker's test (A-D test or K-S test) is high, which indicates that it is hard for the attacker to detect the disturbed message transmission intervals of real events. Second, if the attacker tries to decrease the false negative rate β' by selecting a higher significance level in his distribution test, then the false positive rate α' will increase. As such, no matter which statistical distribution test the attacker uses, there is a tradeoff between false negative rate and false positive rate for the attacker.

We also check the impact of the significance level in our scheme to the detection rate of the attacker. If the significance level α in the A-D test of our scheme is larger, e.g., it is increased from 0.05 to 0.10, then the distributions of message time intervals from real sources in our scheme exhibit less abnormality, i.e., $F(X^i)$ is closer to $F(X)$. Hence, it is harder for the attacker to detect the real events and thus the false negative rate of the attacker is slightly higher (the figure is not shown here because it has only slight difference with Fig. 5).

C. Robustness to Mean Test

After the distribution test, the attacker may conduct a mean test to detect the disturbed means. As message interval data come in continuously, SPRT [14] is a natural choice for such a test. In the SPRT test, after the attacker chooses a threshold ϵ' (in contrast to the corresponding recovery threshold ϵ defined in our scheme), the attacker can do the following to detect real event messages:

- Test two alternatives $H_0 : \mu^i \geq \mu$, $H_1 : \mu^i \leq \mu_1$, if we denote $\mu_1 = (1-\epsilon')\mu$, where μ^i is the sample mean from cell i and μ is the population mean of the exponential distribution. Because in our scheme sample mean tends to be smaller than population mean according to Algorithm 3, with $\mu_i \geq \mu$ the attacker can safely decide that no real event has occurred;
- Choose among three possible decisions: (i) accepting H_0 means that there are no real event messages; (ii) accepting H_1 means that there are real event messages; or (iii) continue the test due to insufficient observations.

Following [14], the above composite hypotheses could be converted to simple hypotheses $H_0 : \mu^i = \mu$ and $H_1 : \mu^i = \mu_1$. Accepting H_0 may cause false negative (β') and accepting H_1 may cause false positive (α').

In more detail, the SPRT mean test for the attacker works as follows. Each time a new message time interval $imd_k^i(k \geq 1)$ from cell i is observed, the following statistics will be calculated

$$s_k = \log \frac{f(imd_1^i, \mu_1) \cdots f(imd_k^i, \mu_1)}{f(imd_1^i, \mu) \cdots f(imd_k^i, \mu)},$$

where f is the probability density function of the exponential distribution. Two boundaries A and B are decided according to the predetermined false positive rate α' and false negative rate β': $A = \log \frac{1-\beta'}{\alpha'}$ and $B = \log \frac{\beta'}{1-\alpha'}$. If $s_k \leq B$, the test is terminated and $H_0 : \mu^i = \mu$ is accepted. If $s_k \geq A$, the test is terminated and $H_1 : \mu^i = \mu_1$ is accepted. If $B < s_k < A$, more observations are needed to make a decision.

Simulation results of the SPRT test for the attacker are presented in Table I and Table II. In both tables, we set the significance level $\alpha = 0.05$ and the recovery threshold $\epsilon = 0.05$ for our scheme; all the generated message transmission intervals in the simulation have passed the exponential distribution test, and about one half of them are actually disturbed ones due to randomly generated real events. In Table I, we fix the attacker's false negative rate β', and check the impact of α' and ϵ' to the number of observations needed for the attacker to make a decision. In Table II, we check the impact of β' and ϵ' under the condition that α' is fixed. From these two tables, we can see that the test results always accept H_0 (theoretically H_1 could also be accepted, subject to the traffic pattern)), which means there is a high chance for the attacker to fail in real event detection.

In addition, there is long delay for the attacker to make a decision. For example, when $\alpha' = 0.05$, $\beta' = 0.20$, and $\epsilon' = 0.05$, after the first message more than 1,000 observations are needed for the attacker to draw a decision. Even if the attacker's conclusion is correct in the end, this may already render the detection worthless.

We further notice from the tables that the number of observations for the attacker to make a decision decreases with the attacker's false positive/negative rate. When the number of observations to make a decision decreases, both the false negative rate β' and false positive rate α' of the attacker become higher. That is, if the attacker wants to make a faster decision, the attacker will have to be willing to accept higher false positive and false negative. Also, the number of observations for the attacker to make a decision decreases with the attacker's recovery threshold ϵ'. If the recovery threshold is higher (e.g., increased from 0.05 to 0.10), the sample data exhibit less abnormality according to the attacker's criteria. Therefore, the attacker draws a quicker conclusion to say that there are no real event messages (although it is still a wrong decision).

In conclusion, the attacker cannot effectively detect the occurrences of real events even after he employs the SPRT-based mean test. We notice that SPRT test is not the only choice for the attacker to detect changed sample mean, but we believe that due to the statistical nature of the problem, the attacker cannot obtain perfect accurate results.

TABLE I
OF OBSERVATIONS TO DRAW A DECISION IN SPRT WHEN α' CHANGES
($\beta' = 0.05$)

α'	0.01	0.05	0.10	0.20	Test result
# of obs. ($\epsilon' = 0.05$)	2198	2192	2058	2054	accept H_0 (false negative)
# of obs. ($\epsilon' = 0.10$)	612	612	611	591	accept H_0 (false negative)

TABLE II
OF OBSERVATIONS TO DRAW A DECISION IN SPRT WHEN β' CHANGES
($\alpha' = 0.05$)

β'	0.01	0.05	0.10	0.20	Test result
# of obs. ($\epsilon' = 0.05$)	3156	2192	1799	1316	accept H_0 (false negative)
# of obs. ($\epsilon' = 0.10$)	921	612	472	361	accept H_0 (false negative)

VI. RELATED WORK

Since Chaum's seminal work in 1981 [15], so far hundreds of papers [3] have been concentrated on building, analyzing, and attacking anonymous communication systems. Due to space limit, we can only discuss those most relevant ones in sensor networks.

In [16], techniques for hiding the base station (message destination) from an external global adversary are studied. In their schemes, every sensor node is a mix and transmits at a constant rate. Different from their work, we are interested in source location privacy. In [2], [17], a random walk based phantom flooding scheme is proposed to defend against an external adversary who attempts to trace back to the data source in a sensor network where sensor nodes report sensing data to a fixed base station. A more recent work [18] proposes a new random walk algorithm. In [19], a path confusion algorithm is proposed to increase source location anonymity. Note that these schemes only works for a local adversary model. In our scheme, we consider a powerful attacker who has the global view of all the network traffic.

In [20], to provide source event unobservability, schemes like ConstRate or ProbRate are used by the sensors. The focus of this work is to reduce the overall network traffic by proactively dropping the dummy messages on their way to the BS. Clearly, this work is complementary to ours and they can be seamlessly integrated to provide both low latency and low communication overhead. In [21], also under the global attacker model, two schemes are proposed. The first one is the ConstRate scheme; the second one is a k-anonymity like source-simulation scheme where $(k - 1)$ fake sources simulate the mobility pattern a mobile real source. All the aforementioned work focuses on providing privacy for outsider attacks. In [22], we proposed techniques to deal with insider attacks by offering different levels of data privacy based on different cryptographic keys for data centric sensor networks.

VII. CONCLUSION

In this paper, after analyzing the source anonymity problem under the global attacker model, we identify the fundamental tradeoff between performance and privacy. For the first time, we propose the notation of statistically strong source anonymity for sensor networks. We also devise a realization scheme called FitProbRate, which achieves statistically strong source anonymity under such a specific circumstance. Performance evaluations demonstrate that by this scheme, the event report latency is largely reduced and source location privacy could be preserved even if the attacker conducts various statistical tests. In our future work, we will investigate different real-world attack models.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, March 2002.
[2] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk, "Enhancing source-location privacy in sensor network routing," *IEEE ICDCS*, 2005.
[3] "The free haven project," http://freehaven.net/anonbib/date.html 2005.
[4] W. Zhang, H. Song, S. Zhu, and G. Cao, "Least privilege and privilege deprivation: Towards tolerating mobile sink compromises in wireless sensor networks," in *Proceedings of ACM Mobihoc*, 2005.
[5] D. Liu, P. Ning, and W. Du, "Attack-resistant location estimation in sensor networks," in *Proceedings of The 4th International Conference on Information Processing in Sensor Networks (IPSN)*, 2005.
[6] A. Back, U. Moller, and A. Stiglic, "Traffic analysis attacks and trade-offs in anonymity providing systems," *The 4th Workshop on Information Hiding (IHW)*, 2001.
[7] A. Pfitzmann and M. Hansen, "Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology," Draft, July 2000.
[8] S. Zhu, S. Setia, and S. Jajodia, "Leap: Efficient security mechanisms for large-scale distributed sensor networks," *ACM CCS*, 2003.
[9] T. W. Anderson and D. A. Darling, "Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes," *The Annals of Mathematical Statistics*, vol. 23, no. 2, pp. 193–212, June 1952.
[10] G. Marsaglia and J. C. W. Marsaglia, "Evaluating the anderson-darling distribution," *Journal of Statistical Software*, vol. 9, no. 2, 2004.
[11] T. W. Anderson and D. A. Darling, "A test of goodness of fit," *Journal of the American Statistical Association*, vol. 49, no. 268, pp. 765–769, December 1954.
[12] M. A. Stephens, "EDF Statistics for Goodness of Fit and Some Comparisons," *Journal of the American Statistical Association*, vol. 69, pp. 730–737, 1974.
[13] J. L. Romeu, "Kolmogorov-simirnov: A goodness of fit test for small samples," *START: Selected Topics in Assurance Related Technologies*, vol. 10, no. 6, 2003.
[14] A. Wald, *Sequential Analysis*. New York: J. Wiley & Sons, 1947.
[15] D. Chaum, "Untraceable electronic mail, return address, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–88, 1981.
[16] J. Deng, R. Han, and S. Mishra, "Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks," *International Conference on Dependable Systems and Networks (DSN'04)*, June 2004.
[17] C. Ozturk, Y. Zhang, and W. Trappe, "Source-location privacy in energy-constrained sensor networks routing," *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04)*, October 2004.
[18] Y. Xi, L. Schwiebert, and W. Shi, "Preserving source location privacy in monitoring-based wireless sensor networks," *the 2nd International Workshop on Security in Systems and Networks(SSN '06)*.
[19] B. Hoh and M. Gruteser, "Protecting location privacy through path confusion," *securecomm*, vol. 0, pp. 194–205, 2005.
[20] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao, "Towards event source unobservability with minimum network traffic in sensor networks," *The ACM Conference on Wireless Network Security (WiSec)*, 2008.
[21] K. Mehta, D. Liu, and M. Wright, "Location privacy in sensor networks against a global eavesdropper," *ICNP*, 2007.
[22] M. Shao, S. Zhu, W. Zhang, and G. Cao, "pDCS: Security and Privacy Support for Data-Centric Sensor Networks," *IEEE INFOCOM*, 2007.