Towards Statistically Strong Source Anonymity for Sensor Networks

YI YANG, Catholic University of America
MIN SHAO, Microsoft
SENCUN ZHU and GUOHONG CAO, The Pennsylvania State University

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 trade-off between security and performance is desirable. In this article, 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 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.

However, the degree of source anonymity in the FitProbRate scheme might decrease as real message rate increases. We propose a *dynamic mean* scheme which has better performance under high real message rates. Simulation results show that the dynamic mean scheme is capable of increasing the attacker's false positive rate and decreasing the attacker's Bayesian detection rate significantly even under high-rate continuous real messages.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms: Security, Design, Algorithms, Performance

Additional Key Words and Phrases: Source anonymity, statistics, privacy, sensor networks

ACM Reference Format:
Yang, Y., Shao, M., Zhu, S., and Cao, G. 2013. Towards statistically strong source anonymity for sensor networks. ACM Trans. Sensor Netw. 9, 3, Article 34 (May 2013), 23 pages.
DOI: http://dx.doi.org/10.1145/2480730.2480737

1. INTRODUCTION

Sensor networks have been envisioned to be very useful for a broad spectrum of emerging civil and military applications [Akyildiz et al. 2002]. However, sensor networks are also confronted with many security threats such as node compromise,

A preliminary version of this work was published in *Proceedings of InfoCom 2008*.
The work of S. Zhu was supported by NSF CAREER 0643906.
Y. Yang is now affiliated with Fontbonne University in St. Louis, MO.
Authors' addresses: Y. Yang (corresponding author) Department of Mathematics and Computer Science, Fontbonne University, 6800 Wydown Blvd., St. Louis, MO 63105-3098; email: yiyangpsu@gmail.com; M. Shao, Microsoft, Redmond, WA 98052; S. Zhu and G. Cao, Department of Computer Science and Engineering, the Pennsylvania State University, University Park, PA 16802.

DOI: http://dx.doi.org/10.1145/2480730.2480737

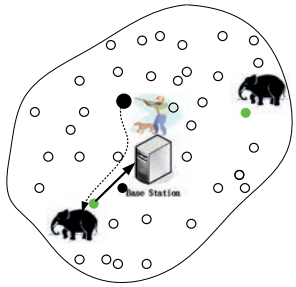

Fig. 1. An application of sensor networks for animal monitoring.

routing disruption, and false data injection, because they normally operate in an unattended, harsh, or hostile environment.

Among all these threats, privacy (especially source anonymity) is of special interest to us 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 Figure 1). When a sensor detects an event, it sends messages including event-related information to the base station. As shown in Kamat et al. [2005], if an attacker (the hunter close to the base station here) can intercept the messages continuously for a certain period of time, he may trace back to the event source in a hop-by-hop fashion. After that, he gets to know the appearance of an endangered animal. Following this, the attacker can take some action to capture or even 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 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 [Kamat et al. 2005], 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, for example, has a hearing range more than three times of the sensors, the capture likelihood is as high as 97%. In addition, a large number of anonymity techniques [Free Haven 2005] 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 article, 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 networkwide 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, for example, 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:

(1) How do we achieve global source anonymity without causing high real event notification latency?
(2) Is it possible to provide perfect global privacy without losing performance benefit?

More specifically, we make the following contributions in this article. 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 *Fitted Probabilistic Rate* (FitProbRate) *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. Third, the source anonymity of the FitProbRate scheme degrades as real message rate increases. We propose a dynamic mean scheme which has better performance under high real message rates.

The rest of the article is organized as follows. We first formalize the problem in Section 2. After that, Section 3 presents the FitProbRate scheme. Its performance is evaluated in Section 4 and its security property is analyzed in Section 5. Then, in Section 6, we introduce the dynamic mean scheme. Finally, we describe the related work in Section 7 and conclude this article in Section 8.

2. PROBLEM FORMALIZATION

2.1. Network Model

As in other sensor networks [Zhang et al. 2005], 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 [Liu et al. 2005]. 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 animal. 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.

2.2. Adversary Model

According to the classification in Back et al. [2001], 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 here.

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 (i.e., not modified or reencrypted) during its forwarding, because given the global view of all the network traffic the adversary is capable of identifying the immediate source of a message and further reconstructing the forwarding path. Third, the attacker may perform 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.

Moreover, the attacker may perform more advanced traffic analysis to find out real source(s). For example, the attacker might grasp statistical tools, such as goodness of fit test, to infer whether time intervals from all sensors follow the same distribution or not. In the long run, the attacker can accumulate enough data and derive the mean of time intervals from every sensor. The attacker might compare whether data flows from certain sensors do not have the same mean. After having the information of means, the attacker can further compare every time interval with the corresponding mean, trying to distinguish relatively small and large intervals and identify patterns from data of time intervals. We assume that the attacker has sufficient resources (e.g., in storage, computation, and communication) to perform the aforesaid attacks.

Note that here we assume the attacker does not go to the suspicious spots for investigation. Because of resource constraints the attacker will choose to go only when his detection rate is high and his false positive rate is low. However, as shown in the simulation of Section 5.4, we can see that the high detection rate of the attacker is at the cost of high false positive rate and low Bayesian detection rate. If the attacker still chooses to physically travel to the identified places under this condition, then he will gain little after a very large quantity of investigations. The detected event might have even completed after such a long delay caused by investigating false alarms.

2.3. Problem Definition

According to Pfitzmann and Hansen [2000], 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 2.1. 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 BS [Zhu et al. 2003]). 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 $X^i = \{imd_1^i, imd_2^i, \ldots\}$. 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 2.2. 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 datasets are statistically indistinguishable only when all these parameters of the probabilistic distribution derived from the two datasets 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 a 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 2.3. (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.

In this definition, although α is not explicitly shown in Conditions 1 and 2, the right-hand side of Condition 1 (i.e., c) is determined by α. Here c is a critical value and α is its corresponding significance level for a statistical hypothesis test. In a statistical hypothesis test, there is normally a constructed table which lists critical values corresponding to different significance level 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 Anderson and Darling [1952] and Marsaglia and Marsaglia [2004]. 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 preceding distance evaluation of CDFs was used in Anderson-Darling test [Anderson and Darling 1954] for goodness of fit tests; therefore, to achieve (α, ϵ)-unobservability our schemes will directly use Anderson-Darling tests. The previous 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.

3. 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.

3.1. Policy for Dummy Traffic Generation

The transmission rate of dummy messages determines the network transmission overhead. As discussed in Section 2.3, 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, for example, 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 ($imd_i, i = 1, 2, \ldots, k-1$) 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 cell. In this way, the attacker cannot predict the next time interval for each cell.

3.2. Policy for Embedding Real Traffic

When a real event happens, by exactly following the ProbRate scheme, that is, 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

ALGORITHM 1: Probabilistic Traffic Generation

Input: mean μ;
Output: a time interval following the exponential distribution with mean μ;
Procedure PTG:
1: seed(*seed*); {Assign *seed* as the seed for random number generation, a unique *seed* is preloaded in each sensor.}
2: return exponential(μ); {Given the mean μ, return a random value following an exponential distribution for message time intervals.}

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 $E_i (1 \leq i \leq k-1)$, the corresponding message should be sent out only when the next time interval (imd_k) and the earlier ones ($imd_i, i = 1, 2, \ldots, k-1$) satisfy the following two conditions:

—The whole series $\{imd_1, imd_2, \ldots, imd_{k-1}, imd_k\}$ 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.

3.2.1. Anderson-Darling Test. Anderson-Darling test [Stephens 1974] (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 follow 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

ALGORITHM 2: Goodness of Fit Test

Input: a sequence of data $\{x_i, 1 \leq i \leq n\}$;
Output: TRUE, if $\{x_i, 1 \leq i \leq n\}$ follows an exponential distribution; FALSE, otherwise.
Procedure Anderson-Darling:
1: sort x_i into an ascending order: $x_1 \leq x_2 \leq \cdots \leq x_n$;
2: calculate the test statistic: A^2;
3: **if** $(A^2 < c)$ {
4: **then** return TRUE;
5: **else** return FALSE;
6: **end if**

Stephens Stephens [1974]. 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, that is, 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 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 $\{imd_1, imd_2, \ldots, imd_{k-1}, imd\}$. 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 dataset, as shown in the algorithm (line 2). Based on our experiments, this can achieve a relatively small

ALGORITHM 3: Search for a Proper Delay to Send a Real Event Message

Input: a sequence of inter-message time intervals $\{imd_i (1 \leq i \leq n)\}$;
Output: a proper delay *imd* to send a real event message;
Procedure search_delay:
1: $\mu = \text{mean}(imd_1, imd_2, \ldots, imd_n)$;
2: *INCREASEPIECE* = rand(0, first quartile);
3: *imd* = *-INCREASEPIECE*;
4: **repeat**
5: **if** imd > upper_bound **then**
6: *INCREASEPIECE* = rand(0, first quartile);
7: *imd* = *-INCREASEPIECE*;
8: **end if**
9: *imd* += *INCREASEPIECE*; {A-D test begins from 0}
10: ret=Anderson-Darling($\{imd_2, imd_3, \ldots, imd_n, imd\}$);
11: **until** ret == TRUE;
12: return *imd*;

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), for example, 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.

3.2.2. 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 (lines 5–8).

From the preceding 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.

ALGORITHM 4: Recovery of Mean

Input: mean μ, a sequence of data $\{x_i, 1 \leq i \leq n\}$;
Output: a proper delay to send out the next message;
Procedure recovery:
1: $sum = \text{sum}(x_2, x_3, \ldots, x_n)$;
2: $dx = \mu\text{-}sum/(n-1)$;
3: $y_1 = (\mu + dx) * n\text{-}sum$;
4: $y_2 = \text{PTG}(\mu)$; {defined in Algorithm 1}
5: **repeat**
6: $x = \text{rand}(y_1, y_2)$;
7: ret = Anderson-Darling($\{x_2, x_3, \ldots, x_n, x\}$);
8: **until** ret == TRUE
9: return x;

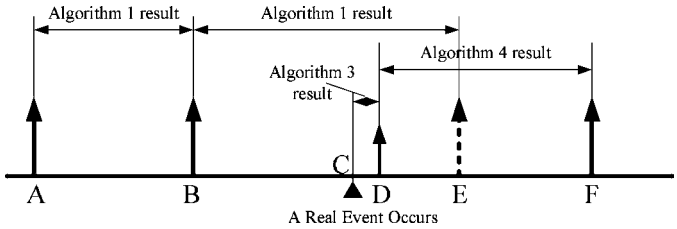

Fig. 2. A running example to illustrate the entire process.

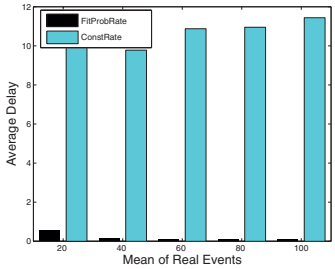

Fig. 3. Comparing average delay in the FitProbRate scheme ($\alpha = 0.05$, $\epsilon = 0.1$) with the ConstRate scheme.

3.3. A Running Example

To illustrate the whole process, a running example is shown in Figure 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.

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.

4. PERFORMANCE EVALUATIONS

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

4.1. 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. Figure 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.

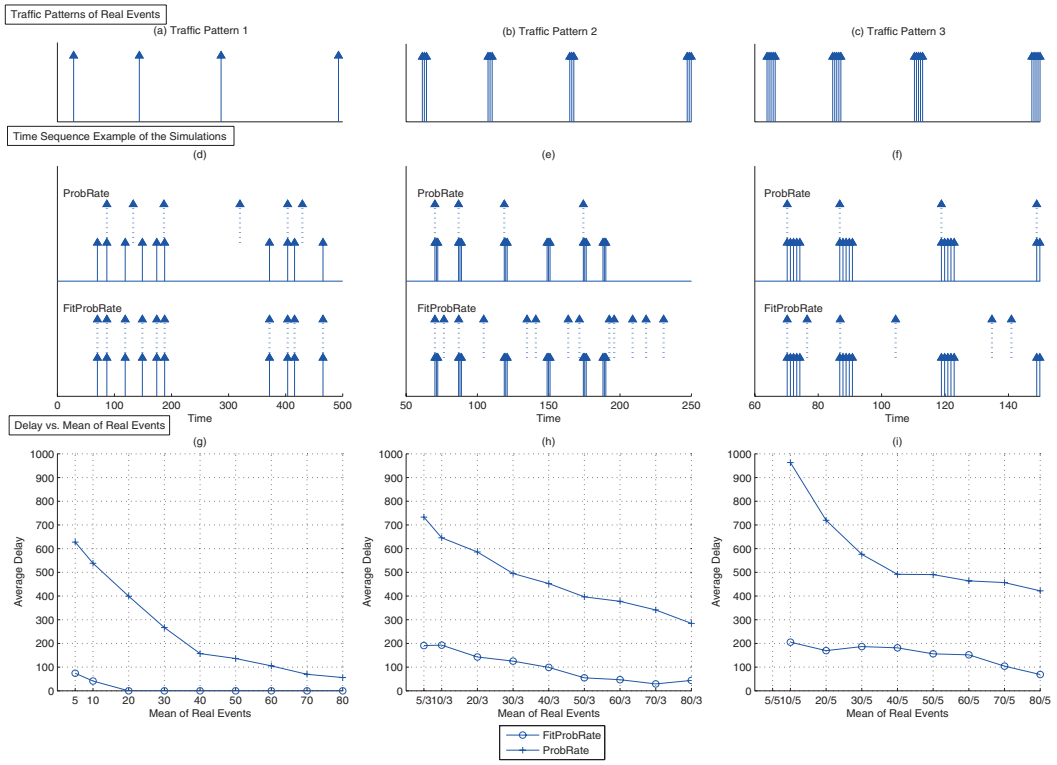

Fig. 4. Performance comparison between the FitProbRate scheme ($\alpha = 0.05$, $\epsilon = 0.1$) and the ProbRate scheme under different real traffic patterns. In (a)–(c), 1, 3, or 5 real event messages are generated in a burst. In (d)–(f) the solid lines are the time points when real events are ready 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.

4.2. 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-s simulation time. For easy illustration, we only show part of the simulation result in Figure 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 Figure 4.

In Figure 4(a), each real event message arrives at the time point according to an exponential distribution; in Figure 4(b) and (c), three and five real event messages are generated in a burst, at the same time points as in Figure 4(a). Figures 4(d) through (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.

Figures 4(g) through (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 Figure 4(i) is also much higher than that of Figure 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.

5. SECURITY ANALYSIS

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

5.1. Security Property

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

THEOREM 5.1. *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, that is, $|\mu^i - \mu| \geq \epsilon$, 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 2.3, 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 shortly 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 [Anderson and Darling 1952] and Kolmogorov-Simirnov (K-S) test [Romeu 2003]. For those distributions that can pass the goodness of fit test, the attacker then further performs the mean test. Those cells whose 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) [Wald 1947] 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. We also check the performance of the FitProbRate scheme under a so-called "01" test.

5.2. 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 as fellows.

—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 trade-off between false positive rate α' and false negative rate β' in the 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 traveling to the fake sources).

Next we use simulations to verify the previous statement. To test false positive, we generate 10,000 groups of pure exponential distributed data. Then we perform K-S and A-D tests 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 5. To test false negative, instead of using pure exponential distributed data, we add real event disturbance into the data and perform the same test.

In Figure 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 trade-off between false negative rate and false positive rate for the attacker.

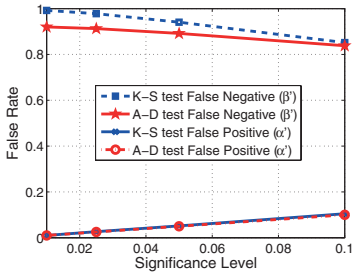

Fig. 5. A trade-off between α' and β' for the attacker ($\alpha = 0.05$).

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, for example, 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, that is, $F(X^i)$ 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 Figure 5).

5.3. 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 [Wald 1947] 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 Wald [1947], the aforesaid 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

Table I. Number 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. Number 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)

$\epsilon = 0.05$ 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.

5.4. Performance under "01" Test

In practice, it is unrealistic to assume that the global attacker has infinite resources to launch attacks. Therefore, we suppose that the attacker maintains a sliding window of most recent $w (w > 0)$ time intervals for each source in the network, in which w is determined by the storage and computation capability of the attacker. Each time when a new time interval from a source is observed, the attacker updates the corresponding sliding window by accepting the new time interval and removing the oldest one from

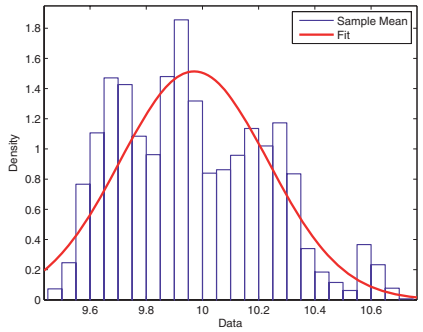

Fig. 6. Sample means fit in a normal distribution with mean of 9.97 and standard deviation of 0.26.

the same source. Based on this, the attacker could continuously calculate a sequence of sample means for each source. Each sample mean is the average of all the time intervals in the sliding window. Because for anonymity purposes the parameter μ of the traffic generator for every source is the same, according to the Central Limit theorem, sample means from all the sources follow a normal distribution with μ as the mean.

We use simulated data (Figure 6) to illustrate this. In the simulation, μ for traffic generator in our system is 10. The attacker's sliding window size is 1,000. From the fitted figure, we can see that sample means follow a normal distribution with mean 9.97 (which is very close to μ) and a small standard deviation 0.26.

That is to say, under the control of parameters α and ϵ, all the sample means are close to each other and also to the parameter of the traffic generator, which is μ. Hence, the population mean from each source (which is the mean of sample means) is close to μ, too. Furthermore, the attacker might observe that some time intervals are smaller than population mean and others are larger than population mean. For simplicity, the attacker could denote the former intervals as "0"s and the latter ones as "1"s. For instance, if a series of time intervals from one sensor is $[0.5, 9.6, 10.7, 15.2, 3.9, 19.4]$ and the mean is 10, then they could be represented as a string "001101". Under this condition, the attacker could identify certain patterns from the data of time intervals and try to distinguish the real source(s) thereafter.

In the FitProbRate scheme, a real message with a time interval smaller than the mean will be represented as "0", which is followed by a recovery time interval that is normally larger than the mean as "1". Then, the attacker is able to identify some "01" patterns and infer that "0"s in these "01" patterns are likely to be real messages. Note the difference between our "01" pattern and the regular string "01". Our "01" pattern means two time intervals: the first time interval is smaller than the mean and the second time interval is larger than the mean. The regular string "01" represents two characters: a character "0" followed by a character "1". We call the attacker's test to identify our "01" patterns (instead of regular string "01") the third type of test from the attacker, which is "01" test.

Case of single message. Here a detection is defined as "a 01 pattern has been identified". Then, detection rate is formulated as follows.

Definition 5.2. Suppose the number of detected "01" patterns caused by real messages is denoted as t and the total number of "01" patterns caused by real messages in the traffic is denoted as $t'(t \leq t')$, then the detection rate of attacker's "01" test is defined as:

$$detection\ rate = t/t'.$$

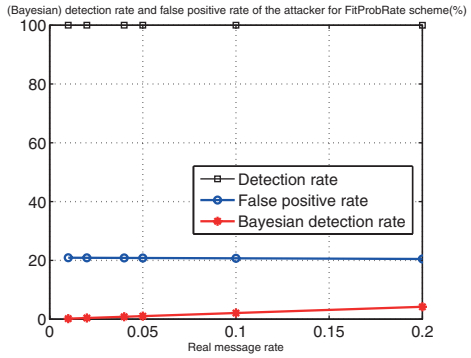

Fig. 7. The attacker's detection rate and false positive rate (in percentage) as a function of real message rate for the FitProbRate scheme under "01" test.

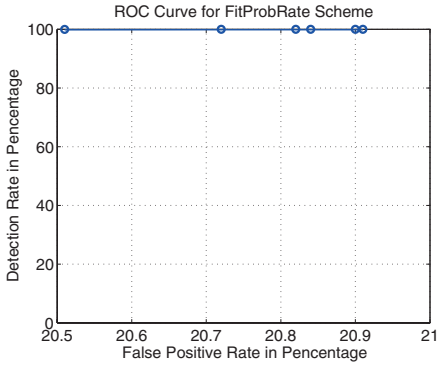

Fig. 8. Receiver Operating Characteristic(ROC) curve for the FitProbRate scheme under "01" test.

Obviously, detection rate is between 0 and 1. Under the "01" test, in normal cases, the attacker has high detection rate, because "01" patterns are easy to detect if sample mean is close to μ.

We use simulations to validate this. In our simulations, there are totally 100 cells in the network and 5 of them are real sources. The goodness of fit test in our simulation is K-S test with significance level of 5%. Window size $w = 1{,}000$. $\mu = 10$. Every point in the figures is averaged over 100 data. As shown in Figure 7 and Figure 8, the attacker's detection rate is as high as 100%. Note that Figure 8 is the Receiver Operating Characteristic (ROC) curve of the attacker, which reflects the relationship between attacker's detection rate and false positive rate.

However, according to base-rate fallacy, with "01" test although an attacker can theoretically capture even 100% real event messages, its false positive rate could also be high, subject to the real event rate. Here false positive rate is formulated as the following.

Definition 5.3. Suppose the total number of detected "01" patterns by the attacker is v, the total number of cells is N, and the number of messages from each cell is m, then the false positive rate of attacker's "01" test is defined as

$$false\ positive\ rate = \frac{v - t}{m \times N - t'}.$$

As shown in the figure, the attacker's false positive rate is as high as 20%.

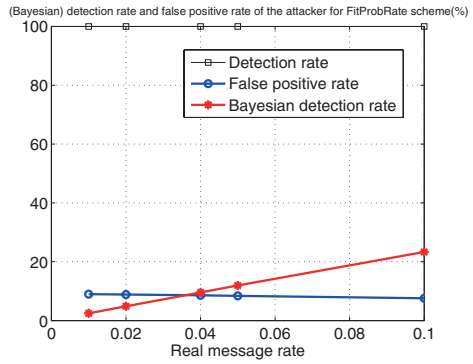

Fig. 9. The attacker's detection rate and false positive rate (in percentage) as a function of real message rate for the FitProbRate scheme under $\lambda = 5$ if attacker tries to detect "01" patterns.

To better understand the effectiveness of the attacker's detection, we check the Bayesian detection rate [Axelsson 1999] of the attacker, which is defined as the probability for an alarm to really indicate a real message. In more detail, we have the following definition.

Definition 5.4. Suppose the total number of detected "01" patterns by the attacker is v and the number of detected "01" patterns caused by real messages is t, then the Bayesian detection rate of attacker's "01" test is defined as

$$Bayesian\ detection\ rate = t/v.$$

From Figure 7, we can see that the attacker's detection is very ineffective because the high detection rate is actually at the cost of high volume of false alarms. When the real message rate is 0.1, the attacker's Bayesian detection rate is only 2.13%, which means the 450 true alarms actually indicate 209,000 false alarms for the attacker.

Case of multiple continuous messages. When a source detects the real event, normally it will send multiple real messages, lasting a duration. According to Kerckhoffs's principle, we assume that the number of real messages that are sent together λ is known to the attacker. For example, λ could equal to 3 or 5. If the real event lasts for a longer time, then λ could be larger. If $\lambda = 3$, the attacker tries to find the match of "000" pattern, because there are three continuous real messages. Similarly, if $\lambda = 5$, the attacker tries to find "00000". We notice that if the attacker does not know the value of this important parameter λ the attacker's detection rate will decrease and the attacker's false positive rate will increase significantly in the following simulations.

As shown in Figure 9 and Figure 10, when the number of continuous real messages sent out together is $\lambda = 5$, the attacker's detection rate is still as high as 100%. However, the attacker's false positive rate decreases from 20% to around 10% and the false positive rate slightly decreases with real message rate. Also, the attacker's Bayesian detection rate increases a lot. When the real message rate is 0.1, this rate increases to above 20%. Apparently, in the FitProbRate scheme, the attacker has better performance if the real message rate is higher or there are multiple continuous real messages sent out together. We need design a scheme that does not have such a limitation.

6. A DYNAMIC MEAN SCHEME

We observe that the attacker has good performance because the sample mean is stable and the attacker's population mean can reflect our system parameter μ accurately. Hence, we construct a dynamic mean scheme by introducing perturbation on the parameter of probabilistic distribution, which is the mean. In this scheme, μ is not a

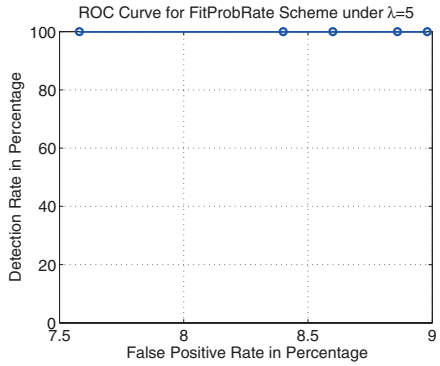

Fig. 10. Receiver Operating Characteristic(ROC) curve for the FitProbRate scheme under $\lambda = 5$ if attacker tries to detect "01" patterns.

fixed parameter. Instead, we have a *disturbance granularity gr*. After each granularity number of message intervals, μ will change to a new value following a uniform distribution in the range of $[a, b](0 < a < b)$ (Algorithm 5: Probabilistic Traffic Generation by Introducing Uniform Distribution). In this case, the attacker may still calculate his own population mean between a and b, but this population mean is not accurate in the sense that it cannot reflect the dynamic nature of the parameter.

ALGORITHM 5: Probabilistic Traffic Generation by Introducing Uniform Distribution

Input: system parameter a and b for uniform distribution, disturbance granularity gr;
Output: time intervals that follow exponential distribution;
Procedure:

```
1: i = 0;
2: loop
3:   if i% gr == 0 then
4:     mu = unifrnd(a, b);
5:   end if
6:   i = i + 1;
7:   return exprnd(mu);
8: end loop
```

For example, after a long time observation, the attacker estimates a population mean of 12. However, the dynamic mean of traffic generator has a parameter of $\mu = 5$ for the next real message. Then, the first real message may be sent out with an interval of 2, the second real interval is 7, and the third one is 10. Thus, overall the observed pattern by the attacker will be "000" instead of "011". In this way, the baseline of mean becomes obscure, so that the attacker's detection becomes inaccurate.

Furthermore, we can postpone the recovery of real messages from the next message interval to a random number of message intervals Δt later. Δt could follow another uniform distribution in the range of $[0, c](c > 0)$. To make sure that the recovery will happen soon, c will not be a very large value.

In the dynamic mean scheme, message time intervals do not follow a specific distribution any more, so that the attacker cannot grasp certain statistic tools, such as goodness of fit test, to detect the disturbance. This actually provides us more flexibility to control and cover real messages' time intervals. In this case, "01" test is still a possible attack. We use simulations to check the performance of this scheme under "01" test and compare it with that of the FitProbRate scheme.

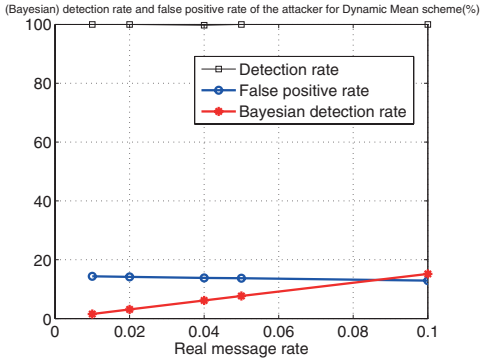

Fig. 11. The attacker's (Bayesian) detection rate and false positive rate (in percentage) as a function of real message rate for the dynamic mean scheme under $\lambda = 5$.

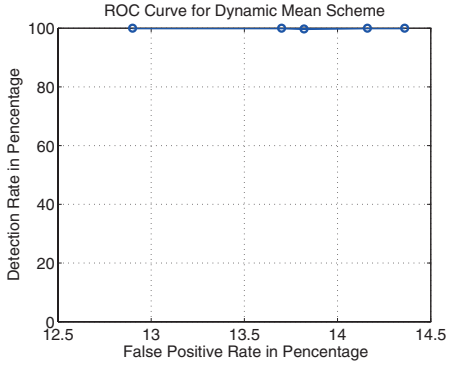

Fig. 12. The attacker's Receiver Operating Characteristic (ROC) curve for the dynamic mean scheme under $\lambda = 5$.

In the simulation, system parameters $a = 5$, $b = 20$, and $c = 5$. The real message duration is $\lambda = 5$. The disturbance granularity is $gr = 10$. From Figure 11 and Figure 12, we can see that the false positive rate of the attacker in the dynamic mean scheme increases and the attacker's Bayesian detection rate decreases, compared with that of the FitProbRate scheme under $\lambda = 5$ (Figure 9 and Figure 10). For example, when real message rate is 0.1, the attacker's false positive rate increases from 7.58% to 12.9% by 70.18% and the attacker's Bayesian detection rate decreases from 23.33% to 15.18% by 34.93%, at the cost of more complexity and higher computational resource consumption in our system.

Degree of anonymity. If the attacker can observe more numbers of pattern "00000" from a cell when $\lambda = 5$, then this cell is more likely to be a real source. For a certain probability, a fake source may happen to have pattern "00000" from its message time intervals; however, real sources have this specific pattern with a higher possibility. Suppose c_i is the count of pattern "00000" for cell i. Then, we can model the probability p_i for a cell being a real source as

$$p_i = \frac{c_i}{\sum_{i=1}^{N} c_i}, \tag{1}$$

in which N is the total number of cells in the network.

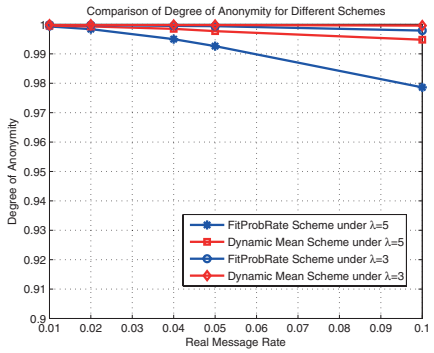

Fig. 13. Comparison of degree of anonymity for different schemes.

According to Díaz et al. [2002], the entropy of a network could be expressed as

$$H(X) = -\sum_{i=1}^{N} p_i \log_2(p_i). \tag{2}$$

The maximum entropy of the network is

$$H_M = \log_2(N). \tag{3}$$

Then, the *degree of anonymity* of the network is defined as

$$d = \frac{H(X)}{H_M}. \tag{4}$$

We use simulations to calculate and compare the degree of anonymity for different schemes. As shown in Figure 13, our system's degree of anonymity decreases with real message rate. Degree of anonymity is higher when the number of continuous messages $\lambda = 3$ than that when $\lambda = 5$. Moreover, our dynamic mean scheme has a higher degree of anonymity compared with the FitProbRate scheme. For example, when the real message rate is 0.1 and $\lambda = 5$, our dynamic mean scheme has a degree of anonymity of 0.9948, whereas the FitProbRate scheme has a degree of anonymity of 0.9786.

7. RELATED WORK

Since Chaum's seminal work in 1981 [Chaum 1981], so far hundreds of papers [Free Haven 2005] have been concentrated on building, analyzing, and attacking anonymous communication systems. Due to space limits, we can only discuss those most relevant ones in sensor networks.

In Deng et al. [2004], 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 Ozturk et al. [2004] and Kamat et al. [2005], 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 [Xi et al. 2006] proposes a new random walk algorithm. In Hoh and Gruteser [2005], a path confusion algorithm is proposed to increase source location anonymity. Note that these schemes only work for a local adversary model. In our scheme, we consider a powerful attacker who has the global view of all the network traffic.

In Yang et al. [2008], 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 Mehta et al. [2007], 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.

Li et al. [2009] give a state-of-the-art survey in privacy preservation techniques for wireless sensor networks. Kamat et al. [2007] introduce buffering delay to provide temporal privacy, which is suitable for delay-tolerant applications of wireless sensor networks. Ouyang et al. [2008] propose four schemes: naive, global, greedy, and probabilistic, to deal with laptop-class attacks. Shao et al. [2009] propose a cross-layer solution in which the event information is first propagated several hops through a MAC-layer beacon. Then, it is propagated at the routing layer to the destination to avoid further beacon delays. To improve source location privacy, Li and Ren [2010] propose dynamic routing schemes, in which messages are first transmitted to randomly selected intermediate nodes to confuse the attacker. Later on, in Pongaliur and Xiao [2011], randomly selected intermediate nodes transform the packets to obfuscate the transmission link from destination to source.

8. CONCLUSION

In this article, after analyzing the source anonymity problem under the global attacker model, we identify the fundamental trade-off 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. We also propose a dynamic mean scheme which has good performance even under continuous real messages with high rates. In our future work, we will investigate various real-world attack models.

REFERENCES

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. 2002. Wireless sensor networks: A survey. *Comput. Netw. 38,* 4.

Anderson, T. W. and Darling, D. A. 1952. Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes. *Ann. Math. Statist. 23,* 2, 193–212.

Anderson, T. W. and Darling, D. A. 1954. A test of goodness of fit. *J. Amer. Statist. Assoc. 49,* 268, 765–769.

Axelsson, S. 1999. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS'99).* 1–7.

Back, A., Miller, U., and Stiglic, A. 2001. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Proceedings of the 4th International Workshop on Information Hiding (IHW'01)*. Springer, 245–257.

Chaum, D. 1981. Untraceable electronic mail, return address, and digital pseudonyms. *Comm. ACM 24,* 2, 84–88.

Deng, J., Han, R., and Mishra, S. 2004. Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks. In *International Conference on Dependable Systems and Networks (DSN'04)*.

Diaz, C., Seys, S., Claessens, J., and Preneel, B. 2002. Towards measuring anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies (PET'02)*. R. Dingledine and P. Syverson, Eds., Lecture Notes in Computer Science, vol. 2482, Springer, 54–68.

Free Haven. 2005. The free haven project. http://freehaven.net/anonbib/date.html.

Hoh, B. and Gruteser, M. 2005. Protecting location privacy through path confusion. In *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm'05).* 194–205.

KAMAT, P., XU, W. Y., TRAPPE, W., AND ZHANG, Y. 2007. Temporal privacy in wireless sensor networks. In *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS'07)*.

KAMAT, P., ZHANG, Y., TRAPPE, W., AND OZTURK, C. 2005. Enhancing source-location privacy in sensor network routing. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*. IEEE Computer Society, Los Alamitos, CA, 599–608.

LI, N., ZHANG, N., DAS, S. K., AND THURAISINGHAM, B. 2009. Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Netw. 7,* 8, 1501–1514.

LI, Y. AND REN, J. 2010. Source-location privacy through dynamic routing in wireless sensor networks. In *Proceedings of the 29th Conference on Information Communications (INFOCOM'10)*. 2660–2668.

LIU, D., NING, P., AND DU, W. 2005. Attack-resistant location estimation in sensor networks. In *Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN'05)*.

MARSAGLIA, G. AND MARSAGLIA, J. C. W. 2004. Evaluating the anderson-darling distribution. *J. Statist. Softw. 9,* 2.

MEHTA, K., LIU, D., AND WRIGHT, M. 2007. Location privacy in sensor networks against a global eavesdropper. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP'07)*. 314–323.

OUYANG, Y., LE, Z., LIU, D., FORD, J., AND MAKEDON, F. 2008. Source location privacy against laptop-class attacks in sensor networks. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks (SecureComm'08)*.

OZTURK, C., ZHANG, Y., AND TRAPPE, W. 2004. Source-location privacy in energy-constrained sensor networks routing. In *Proceedings of tlhe ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04)*.

PFITZMANN, A. AND HANSEN, M. 2000. Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology. http://www.caida.org/publications/bib/networking/entries/pfitzmann_terminology.xml.

PONGALIUR, K. AND XIAO, L. 2011. Maintaining source privacy under eavesdropping and node compromise attacks. In *Proceedings of the 30th International Conference on Computer Communications (INFOCOM'11)*.

ROMEU, J. L. 2003. Kolmogorov-simirnov: A goodness of fit test for small samples. *START: Select. Topics Assur. Related Technol. 10,* 6.

SHAO, M., HU, W., ZHU, S., CAO, G., KRISHNAMURTHY, S., AND PORTA, T. L. 2009. Cross-layer enhanced source location privacy in sensor networks. In *Proceedings of the 6th Annual IEEE Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON'09)*.

STEPHENS, M. A. 1974. EDF statistics for goodness of fit and some comparisons. *J. Amer. Statist. Assoc. 69*, 730–737.

WALD, A. 1947. *Sequential Analysis*. J. Wiley and Sons, New York.

XI, Y., SCHWIEBERT, L., AND SHI, W. 2006. Preserving source location privacy in monitoring-based wireless sensor networks. In *Proceedings of the 2nd International Workshop on Security in Systems and Networks (SSN'06)*.

YANG, Y., SHAO, M., ZHU, S., URGAONKAR, B., AND CAO, G. 2008. Towards event source unobservability with minimum network traffic in sensor networks. In *Proceedings of the ACM Conference on Wireless Network Security (WiSec)*.

ZHANG, W., SONG, H., ZHU, S., AND CAO, G. 2005. Least privilege and privilege deprivation: Towards tolerating mobile sink compromises in wireless sensor networks. In *Proceedings of the ACM International Symposium Mobile Ad Hoc Networkiing and Computing (MobiHoc'05)*.

ZHU, S., SETIA, S., AND JAJODIA, S. 2003. Leap: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*.

Received June 2011; revised March 2012; accepted April 2012