Temporal Privacy in Wireless Sensor Networks: Theory and Practice

PANDURANG KAMAT, WENYUAN XU, WADE TRAPPE, and YANYONG ZHANG
Rutgers University

Although the content of sensor messages describing "events of interest" may be encrypted to provide confidentiality, the context surrounding these events may also be sensitive and therefore should be protected from eavesdroppers. An adversary armed with knowledge of the network deployment, routing algorithms, and the base-station (data sink) location can infer the temporal patterns of interesting events by merely monitoring the arrival of packets at the sink, thereby allowing the adversary to remotely track the spatio-temporal evolution of a sensed event. In this paper we introduce the problem of temporal privacy for delay-tolerant sensor networks, and propose adaptive buffering at intermediate nodes on the source-sink routing path to obfuscate temporal information from the adversary. We first present the effect of buffering on temporal privacy using an information-theoretic formulation, and then examine the effect that delaying packets has on buffer occupancy. We observe that temporal privacy and efficient buffer utilization are contrary objectives, and then present an adaptive buffering strategy that effectively manages these tradeoffs. Finally, we evaluate our privacy enhancement strategies using simulations, where privacy is quantified in terms of the adversary's mean square error.

Categories and Subject Descriptors: D.4.6 [**Operating Systems**]: Security and Protection

General Terms: Security, Algorithms, Design, Measurement, Performance, Theory

Additional Key Words and Phrases: Sensor networks, privacy, security, temporal privacy

ACM Reference Format:

Kamat, P., Xu, W., Trappe, W., and Zhang, Y. 2009. Temporal privacy in wireless sensor networks: Theory and practice. ACM Trans. Sens. Netw. 5, 4, Article 28 (November 2009), 24 pages. DOI = 10.1145/1614379.1614380 http://doi.acm.org/10.1145/1614379.1614380.

1. INTRODUCTION

Sensor networks are being deployed to monitor a vast array of phenomena. The information surrounding these measurements can have varying levels of importance, and for this reason conventional security services, such as encryption and authentication, have been migrated to the sensor domain [Perrig et al. 2002; Bohge and Trappe 2003; Karlof and Wagner 2003; Zhu et al. 2003; Chan

Authors' address: Wireless Information Network Laboratory (WINLAB), Rutgers University, 671 Route 1 South, North Brunswick, NJ 08902; email: pkamat@winlab.rutgers.edu.

DOI 10.1145/1614379.1614380 http://doi.acm.org/10.1145/1614379.1614380

and Perrig 2003]. However, in spite of the protection that such operations might provide, there are many aspects associated with the creation and delivery of sensor messages that remain unprotected by conventional security mechanisms, and such contextual information should be protected using complimentary techniques.

Since wireless sensor networks employ a broadcast medium, an adversary may monitor sensor communications to piece together knowledge of the context surrounding sensor messages. In particular, by applying wireless localization algorithms and some level of diligence, an adversary will be able to infer the sensor network deployment, that is, an association of sensor IDs with their physical locations. This information, combined with knowledge of the routing algorithms employed and the location of the base-station (data sink), can allow the adversary to track the spatio-temporal evolution of a sensed-event from the remote location of the network sink by merely monitoring the arrival of incoming packets [Tseng and Lin 2005]. This spatio-temporal information is available regardless of whether the adversary can decipher encrypted packet payloads, and represents a breach of the spatio-temporal privacy associated with the sensor network's operation. This breach of privacy can be put to very malicious use. For example, in an asset tracking sensor network, an adversary can use the spatio-temporal characteristics of the network traffic to determine the speed and direction of motion of an asset and track it down.

In order to protect against such a privacy breach, there are two types of information that can be protected: the spatial information surrounding the flow of sensor messages, and the temporal context surrounding the creation of sensor readings. Protecting the spatial context of sensor routing involves obscuring the location of the source sensor [Kamat et al. 2005; Ozturk et al. 2004], as well as the location of the network sink [Deng et al. 2004, 2005]. However, should an adversary compromise the defense mechanisms meant to protect a sensor network's spatial context and learn the location of the originating sensor and the network sink, then the spatio-temporal context of a sensor's message flow may still be protected by employing mechanisms that protect the *temporal context* of the sensor's message.

In this article we focus on the problem of protecting the temporal context associated with a sensor's measurement of underlying physical phenomena. Specifically, for the typical delay-tolerant application, we propose the use of additional store-and-forward buffering at intermediate nodes along the routing path between a source sensor and the sink in order to obfuscate the time of creation associated with the flow of sensor messages.

We begin the article in Section 2 by describing our sensor network model, overview the problem of temporal privacy and how additional buffering can enhance privacy. We then examine the two conflicting aspects of buffering: in Section 3, we formulate temporal privacy from an information-theoretic perspective, and in Section 4, we examine the stress that additional delay places on intermediate buffers. Then, in Section 5, we present an adaptive buffering strategy that effectively manages these tradeoffs through the preemptive release of packets as buffers attain their capacity. We evaluate our temporal privacy solutions in Section 6 through simulations involving a large-scale network, where the adversary's mean square error is used to quantify the temporal privacy. Finally, we present related work in Section 7, and conclude the article in Section 8.

2. OVERVIEW OF TEMPORAL PRIVACY IN SENSOR NETWORKS

We start our overview by describing a couple of scenarios that illustrate the issues associated with temporal privacy. To begin, consider a sensor network that has been deployed to monitor an animal habitat [Kamat et al. 2005; Szewczyk et al. 2004]. In this scenario, animals ("assets") move through the environment, their presence is sensed by the sensor network and reported to the network sink. The fact that the network produces data and sends it to the sink provides an indication that the animal was present at the source at a specific time. If the adversary is able to associate the origin time of the packet with a sensor's location, then the adversary will be able to track the animal's behavior—a dangerous prospect if the animal is endangered and the adversary is a hunter! This same scenario can be easily translated to a tactical environment, where the sensor network monitors events in support of military networked operations. In asset tracking, if we add temporal ambiguity to the time that the packets are created then, as the asset moves, this would introduce spatial ambiguity and make it harder for the adversary to track the asset.

The situations where temporal privacy is important are not always associated with protecting spatio-temporal context, but instead there are scenarios where we are solely interested in masking the time at which an event occurred. For example, sensor networks may be deployed to monitor inventory in a warehouse. In this scenario, a sensor would create audit logs associated with the removal/relocation of items (bearing RFID tags) within the warehouse and route these audit messages to the network sink. Here, an adversary located near the sink (perhaps outside the warehouse) could observe packets arriving and use this information to infer the stock levels or the volume of transactions going through a warehouse at a specific time. Such information could be of great benefit to a rival corporation that is interested in knowing its competitor's sales and inventory profile. Here, if we add temporal ambiguity to the delivery of the audit messages, then the warehouse would still be able to verify its inventory against purchase orders, but the competitor would have outdated information about the inventory activity.

For both scenarios, temporal privacy amounts to preventing an adversary from inferring the time of creation associated with one or more sensor packets arriving at the network sink. In order to protect the temporal context of the packet's creation, it is possible to introduce additional, random delay to the delivery of packets in order to mask a sensor reading's time of creation. However, although delaying packets might increase temporal privacy, this strategy also necessitates the use of buffering within the network and places new stress on the internal store-and-forward network buffers.

We may define a generic model for both the sensor network and the adversary that captures the most relevant features of the temporal privacy problem in this article. The abstract sensor network model that we will use involves:

—*Delay-Tolerant Application*. A sensor application that is delay-tolerant in the sense that observations can be delayed by reasonable amounts of time before arriving at the monitoring application, thereby allowing us to introduce additional delay in packet delivery.

—*Payload Encrypted*. The payload contains application-level information, such as the sensor reading, application sequence number and the time-stamp associated with the sensor reading. In order to guarantee the confidentiality of this data, conventional encryption is employed.

—*Headers are Cleartext*. The headers associated with essential network functionality are not encrypted. For example, the routing header associated with Woo et al. [2003], and used in the TinyOS 1.1.7 release (described in `MultiHop.h`) includes the ID of the previous hop, the ID of the origin (used in the routing layer to differentiate between whether the packet is being generated or forwarded), the routing-layer sequence number (used to avoid loops, not flow-specific and hence cannot help the adversary in estimating time of creation), and the hop count.

On the other hand, the assumptions that we have for the adversary are

—*Protocol-Aware*. By Kerckhoff's Principle [Trappe and Washington 2002], we assume the adversary has knowledge of the networking and privacy protocols being employed by the sensor network. In particular, the adversary knows the delay distributions being used by each node in the network.

—*Able to Eavesdrop*. We assume that the adversary is able to eavesdrop on communications in order to read packet headers, or control traffic. We emphasize that the adversary is not able to decipher packet contents by decrypting the payloads, and hence the adversary must infer packet creation times solely from network knowledge and the time it witnesses a packet.

—*Deployment-Aware*. We assume that the adversary at the sink and is aware of the identity of all sensor nodes. Since the adversary can monitor communications, we assume that the adversary knows the source identity associated with each transmission. Further, since the adversary is aware of the routing protocols employed and can eavesdrop, the adversary is able to build its own source-sink routing tables.

—*Nonintrusive*. The adversary does not interfere with the proper functioning of the network, otherwise intrusion detection measures might flag the adversary's presence. In particular, the adversary does not inject or modify packets, alter the routing path, or destroy sensor devices.

Taken together, we note that we have separated out issues associated with obscuring the location of the source's origin, and solely focus on temporal privacy. We note, however, that in practice the combination of temporal privacy methods with location-privacy methods will yield a more complete solution to protecting contextual privacy in sensor networks.

3. TEMPORAL PRIVACY FORMULATION

We start by first examining the theoretical underpinnings of temporal privacy. Our discussion will start by first setting up the formulation using a simple network of two nodes transmitting a single packet, and then we extend the formulation to more general network scenarios.

3.1 Temporal Privacy: Two-Party Single-Packet Network

We begin by considering a simple network consisting of a source S, a receiver node R, and an adversarial node E that monitors traffic arriving at R. The goal of preserving temporal privacy is to make it difficult for the adversary to infer the time when a specific packet was created. Suppose that the source sensor S observes a phenomena and creates a packet at some time X. In order to obfuscate the time at which this packet was created, S can choose to locally buffer the packet for a random amount of time Y before transmitting the packet. Disregarding the negligible time it takes for the packet to traverse the wireless medium, both R and E will witness that the packet arrives at a time $Z = X + Y$. The legitimate receiver can decrypt the payload, which contains a timestamp field describing the correct time of creation. The adversary's objective is to infer the time of creation X, and since it cannot decipher the payload, it must make an inference based solely upon the observation of Z and (by Kerckhoff's Principle) knowledge of the buffering strategy employed at S.

The ability of E to infer X from Z is controlled by two underlying distributions: first is the a priori distribution $f_X(x)$, which describes the knowledge the adversary had for the likelihood of the message creation prior to observing Z; and second, the delay distribution $f_Y(y)$, which the source employs to mask X. The amount of information that E can infer about X from observing Z is measured by the mutual information:

$$\begin{aligned} I(X;Z) &= h(X) - h(X|Z) = h(Z) - h(Z|X) && (1)\\ &= h(Z) - h(Y), && (2) \end{aligned}$$

where $h(X)$ is the differential entropy of X. For certain choices of f_X and f_Y, we may directly calculate $I(X;Z)$. For example, if $X \sim Exp(\lambda)$ (i.e., exponential with mean $1/\lambda$), and $Y \sim Exp(\lambda)$, then $Z \sim Erlang(2, \lambda)$, and $h(Z) = -\psi(2) + \ln\Gamma(2) - \ln(\lambda) + 2$, where $\psi(w)$ is the digamma function and $\Gamma(w)$ is the gamma function. For this case, $h(Y) = 1 - \ln\lambda$, and hence $I(X;Z) = 1 - \psi(2) \approx 1.077$. In other words, roughly 1 nat of information about X is learned by observing Z. For more general distributions, the entropy-power inequality [Cover and Thomas 1991] gives a lower bound

$$I(X;Z) \geq \frac{1}{2\ln 2}\left(2^{2h(X)} + 2^{2h(Y)}\right) - h(Y). \qquad (3)$$

In general, however, the distribution for X is fixed and determined by an underlying physical phenomena being monitored by the sensor. Since the objective of the temporal privacy-enhancing buffering is to hide X, we may formulate the temporal privacy problem as

$$\min_{f_Y(y)} I(X;Z) = h(X+Y) - h(Y),$$

or in other words, choose a delay distribution f_Y so that the adversary learns as little as possible about X from Z.[1]

3.2 Temporal Privacy: Two-Party Multiple-Packet Network

We now extend the formulation of temporal privacy to the more general case of a source S sending a stream of packets to a receiver R in the presence of an adversary E. In this case, the sender S will create a stream of packets at times $X_1, X_2, \ldots, X_n, \ldots$, and will delay their transmissions by $Y_1, Y_2, \ldots, Y_n, \ldots$. The packets will be observed by E at times $Z_1, Z_2, \ldots, Z_n, \ldots$. In going to the more general case of a packet stream, several new issues arise. First, as noted earlier in Section 2, when we delay multiple packets it will be necessary to buffer these packets. For now we will hold off on discussing queuing issues until Section 4. The next issue involves how the packets should be delayed. There are many possibilities here. For example, one possibility would have packets released in the same order as their creation, that is, $Z_1 < Z_2 < \ldots < Z_n$, which would correspond to choosing Y_j to be at least the wait time needed to flush out all previous packets. Such a strategy does not reflect the fact that most sensor monitoring applications do not require that packet ordering is maintained. Therefore, a more natural delay strategy would involve choosing Y_j independent of each other and independent of the creation process $\{X_j\}$. Consequently, there will not be an ordering of $(Z_1, Z_2, \ldots, Z_n, \ldots)$.

In our sensor network model, however, we assumed that the sensing application's sequence number field was contained in the encrypted payload, and consequently the adversary does not directly observe $(Z_1, Z_2, \ldots, Z_n, \ldots)$, but instead observes the sorted process $\{\tilde{Z}_j\} = \Upsilon(\{Z_j\})$, where $\Upsilon(\{Z_j\})$ denotes the permutations needed to achieve a temporal ordering of the elements of the process $\{Z_j\}$, that is, $\{\tilde{Z}_j\} = (\tilde{Z}_1, \tilde{Z}_2, \ldots, \tilde{Z}_n, \ldots)$ where $\tilde{Z}_1 < \tilde{Z}_2 < \cdots$. The adversary's task thus becomes inferring the process $\{X_j\}$ from the sorted process $\{\tilde{Z}_j\}$. The amount of information gleaned by the adversary after observing $\tilde{Z}^n = (\tilde{Z}_1, \ldots, \tilde{Z}_n)$ is thus $I(X^n; \tilde{Z}^n)$, and the temporal-privacy objective of the system designer is to make $I(X^n; \tilde{Z}^n)$ small.

Although it is analytically cumbersome to access $I(X^n; \tilde{Z}^n)$, we may use the data processing inequality [Cover and Thomas 1991] on $X^n \to Z^n \to \tilde{Z}^n$ to obtain the relationship $0 \le I(X^n, \tilde{Z}^n) \le I(X^n, Z^n)$, which allows us to use $I(X^n, Z^n)$ in a pinching argument to control $I(X^n, \tilde{Z}^n)$. Expanding $I(X^n, Z^n)$ as

$$\begin{aligned} I(X^n, Z^n) &= h(Z^n) - h(Y^n) && (4)\\ &\le \sum_{j=1}^{n} \big(h(Z_j) - h(Y_j)\big) && (5)\\ &= \sum_{j=1}^{n} I(X_j, Z_j), && (6)\end{aligned}$$

[1]The astute reader will note the similarity with the information-theoretic formulation of communication, where the objective is to maximize mutual information.

we may thus bound $I(X^n, Z^n)$ using the sum of individual mutual information terms.

As before, the objective of temporal privacy enhancement is to minimize the information that the adversary gains, and hence to mask $\{X_j\}$, we should minimize $I(X^n, Z^n)$. Although there are many choices for the delay process $\{Y_j\}$, the general task of finding a nontrivial stochastic process $\{Y_j\}$ that minimizes the mutual information for a specific temporal process $\{X_j\}$ is challenging and further depends on the sensor network design constraints (e.g. buffer storage). In spite of this, however, we may seek to optimize within a specific type of process $\{Y_j\}$, and from this make some general observations.

As an example of this, let us look at an important and natural example. Suppose that the source sensor creates packets at times $\{X_j\}$ as a Poisson process of rate λ, that is, the interarrival times A_j are exponential with mean $1/\lambda$, and that the delay process $\{Y_j\}$ corresponds to each Y_j being an exponential delay with mean $1/\mu$. One motivation for choosing an exponential distribution for the delay is the well-known fact that the exponential distribution yields maximal entropy for nonnegative distributions. We note that $X_j = \sum_{k=1}^{j} A_k$ (and hence the X_j are j-stage Erlangian random variables with mean j/λ). Using the result of Theorem 3(d) from Anantharam and Verdu [1996], we have that

$$\begin{aligned} I(X_j; Z_j) &= I(X_j; X_j + Y_j) \\ &= \ln\left(1 + \frac{j\mu}{\lambda}\right) - D\left(f_{X_j+Y_j} \| f_{\overline{X}_j+Y_j}\right) \\ &\leq \ln\left(1 + \frac{j\mu}{\lambda}\right). \end{aligned}$$

Here, the $D(f\|g)$ corresponds to the divergence between two distributions f and g, while $\overline{X}$ is the mixture of a point mass and exponential distribution with the same mean as X, as introduced in Anantharam and Verdu [1996]. Since divergence is nonnegative and we are only interested in pinching $I(X^n; \tilde{Z}^n)$, we may discard this auxiliary term. Using that result, we have that

$$I(X^n, Z^n) \leq \sum_{j=1}^{n} \ln\left(1 + \frac{j\mu}{\lambda}\right). \tag{7}$$

Our objective is to make

$$0 \leq I(X^n; \tilde{Z}^n) \leq I(X^n, Z^n) \leq \sum_{j=1}^{n} \ln\left(1 + \frac{j\mu}{\lambda}\right)$$

small, and from this we can see that by tuning μ to be small relative to λ (or equivalently, the average delay time $1/\mu$ to be large relative to the average interarrival time $1/\lambda$), we can control the amount of information the adversary learns about the original packet creation times. It is clear that choosing μ too small will place a heavy load on the source's buffer. We will revisit buffer issues in Section 4 and Section 5.

3.3 Temporal Privacy: Multihop Networks

In the previous subsection, we considered a simple network case consisting of two nodes, where the source performs all of the buffering. More general sensor networks consist of multiple nodes that communicate via multihop routing to a sensor network sink. For such networks, the burden of obfuscating the times at which a source node creates packets can be shared among other nodes on the path between the source and the sensor network sink.

To explain, we may consider a generic sensor network consisting of an abundant supply of sensor nodes, and focus on an N-hop routing path between the source and the network sink. By doing so, we are restricting our attention to a line-topology network $S \to F_1 \to F_2 \to \cdots \to F_{N-1} \to R$, where R denotes the receiving network sink, and F_j denotes the j-th intermediate node on the forwarding path.

By introducing multiple nodes, the delay process $\{Y_j\}$ can be decomposed across multiple nodes as

$$Y_j = Y_{0j} + Y_{1j} + \cdots + Y_{N-1,j},$$

where Y_{kj} denotes the delay introduced at node k for the j-th packet (we use Y_{0j} to denote the delay used by the source node S). Thus, each node k will buffer each packet j that it receives for a random amount of time Y_{kj}.

This decomposition of the delay process $\{Y_j\}$ into subdelay processes $\{Y_{kj}\}$ allows for great flexibility in achieving both temporal privacy goals and ensuring suitable buffer utilization in the sensor network. For example, it is well known that traffic loads in sensor networks accumulate near network sinks, and it may be possible to decompose $\{Y_j\}$ so that more delay is introduced when a forwarding node is further from the sink.

4. QUEUING ANALYSIS OF PRIVACY-ENHANCING BUFFERING

Although delaying packets might increase temporal privacy, such a strategy places a burden on intermediate buffers. In this section we will examine the underlying issues of buffer utilization when employing delay to enhance temporal privacy.

When using buffering to enhance temporal privacy, each node on the routing path will receive packets and delay their forwarding by a random amount of time. As a result, sensor nodes must buffer packets prior to releasing them, and we may formulate the buffer occupancy using a queuing model. In order to start our discussion, let us again examine the simple two-node case where a source node S generates packets according to an underlying process and the packets are delayed according to an exponential distribution with average delay $1/\mu$, prior to being forwarded to the receiver R, as depicted in Figure 1(a). If we assume that the creation process is Poisson with rate λ (if the process is not Poisson, the source may introduce additional delay to shape the traffic), then the buffering process can be viewed as an $M/M/\infty$ queue where, as new packets arrive at the buffer, they are assigned to a new "variable-delay server" that processes each packet according to an exponential distribution with mean $1/\mu$. Following the standard results for $M/M/\infty$ queues, we have that the amount

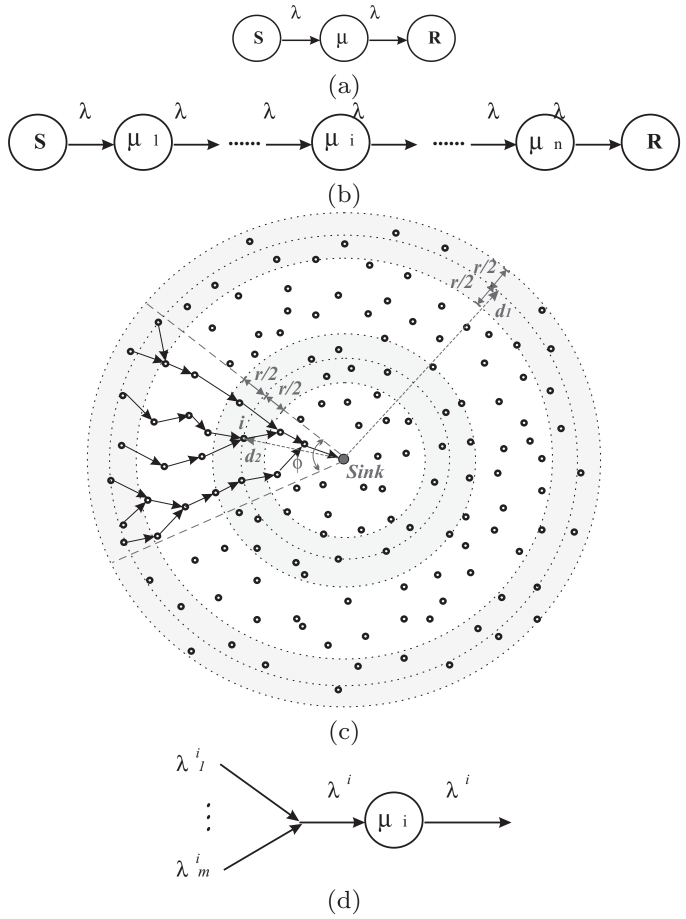


Fig. 1. (a) Queue model for buffering at the source node S, (b) chain of queues along a routing path from S to receiver sink R, (c) the effect of flow convergence in a large sensor network, and (d) queuing model for the merging of traffic flows at an intermediate sensor node.

of packets being stored at an arbitrary time, $N(t)$, is Poisson distributed, with $p_k = P\{N(t) = k\} = \frac{\rho^k}{k!}e^{-\rho}$, where $\rho = \lambda/\mu$ is the system utilization factor. $\overline{N}$, the expected number of messages buffered at S, is ρ.

The slightly more complicated scenario involving more than one intermediate node allows for the buffering responsibility to be divided across the routing path, and is depicted by a chained path in Figure 1(b). A tandem queuing network is formed, where a message departing from node i immediately enters an $M/M/\infty$ queue at node $i+1$. Thus, the interdeparture times from the former generate the interarrival times to the latter. According to Burke's Theorem [Bertsekas and Gallager 1992], the steady-state output of a stable $M/M/m$ queue with input parameter λ and service-time parameter μ for each of the m servers is in fact a Poisson process at the same rate λ when $\lambda < \mu$. Hence, we may generally model each node i on the path as an $M/M/\infty$ queue with average input message rate

λ, but with average service-time $1/\mu_i$ (to allow each node to follow its own delay distribution).

So far we have only considered a single routing path in a sensor network, but in practice the network will monitor multiple phenomena simultaneously, and consequently there will be multiple source-sink flows traversing the network. As a result, for the most general scenario, the topological structure of the network will have an impact on buffer occupancy. For example, nodes that are closer to network sink typically have higher traffic loads, and thus will be expected to suffer from a higher buffer occupancy than nodes further from the sink. We now explore this behavior, and the relationship between buffering for privacy-enhancement and the traffic load placed on intermediate nodes due to flow convergence in the sensor network.

Consider a sensor network deployment as depicted in Figure 1(c), where we have assumed (without loss of generality) that there is only one sink. Here, multiple sensors generate messages intended for the sink, and each message is routed in a hop-by-hop manner based on a routing tree (as suggested in the figure). Message streams merge progressively as they approach the sink. If we assume that the senders in the network generate Poisson flows, then by the superposition property of Poisson processes, the combined stream arriving at node i of m independent Poisson processes with rate λ^i_j is a Poisson process with rate $\lambda^i = \lambda^i_1 + \lambda^i_2 + \cdots + \lambda^i_m$. We depict this phenomena for node i in Figure 1(d), where m is the number of "routing" children for node i. Additionally, we let $1/\mu_i$ be the average buffer delay injected by node i. Then node i is an $M/M/\infty$ queue, with arrival parameter λ^i and departure parameter μ_i, yielding:

—$N_i(t)$, the number of packets in the buffer at node i, is Poisson distributed.

—$p_{ik} = P\{N_i(t) = k\} = \frac{\rho_i^k}{k!}e^{-\rho_i}$, where $\rho_i = \lambda^i/\mu_i$.

—The expected number of messages at node i is $\overline{N_i} = \rho_i$.

As expected, if we choose our delay strategy at node i such that μ_i is much smaller than λ^i (as is desirable for enhanced temporal privacy), then the expected buffer occupancy $\overline{N_i}$ will be large. Thus, temporal privacy and buffer utilization are conflicting system objectives.

We now evaluate the impact of the depth of node i in the routing tree (the number of hops from the node i to the sink). For the sake of calculations, we shall assume that the density η of the sensor deployment is sufficient that a communicating sensor node will always find a path to the network sink. Additionally, let us denote the average geographical distance between parents and children in the routing tree by r. Then, to quantify the effect of flow convergence on the local traffic rate in the sensor network, let us assume that an outer annulus O_1 of distance d_1, angular spread φ and width r creates a total traffic of rate λ_{O_1} packets/second, as depicted in Figure 1(c). Hence, in a spatial ensemble sense, each node carries an average traffic rate of $\overline{\lambda_{O_1}} = \lambda_{O_1}/(\varphi r d_1 \eta)$. This traffic flows toward the sink, and if we examine an annulus at distance $d_2 < d_1$ with width r and spread φ, the area of this annulus is $\varphi r d_2$, and there will be an average of $\varphi r d_2 \eta$ sensors in O_2 carrying a total rate of λ_{O_1}. Hence, on average, each sensor in this inner annulus will carry traffic of rate $\overline{\lambda_{O_2}} = \lambda_{O_1}/(\varphi r d_2 \eta)$. Comparing the average traffic load $\overline{\lambda_{O_2}}$ that a single sensor in an inner annulus O_2 carries with the average traffic load $\overline{\lambda_{O_1}}$ of a single sensor in annulus O_1, yields

$$\frac{\overline{\lambda_{O_2}}}{\overline{\lambda_{O_1}}} = \frac{d_1}{d_2}, \tag{8}$$

and hence traffic load increases in inverse relationship to the distance a node is from the sink.

The last issue that we need to consider is the amount of storage available for buffering at each sensor. As sensors are resource-constrained devices, it is more accurate to replace the $M/M/\infty$ queues with $M/M/k/k$ queues, where memory limitations imply that there are at most k servers/buffer slots, and each buffer slot is able to handle 1 message. If an arriving packet finds all k buffer slots full, then either the packet is dropped or, as we shall describe in Section 5, a preemption strategy can be employed. For now, we just consider packet dropping. We note that packet dropping at a single node causes the outgoing process to lose its Poisson characteristics. However, we further note that by Kleinrock's Independence approximation (the merging of several packet streams has an affect akin to restoring the independence of interarrival times) [Bertsekas and Gallager 1992], we may continue to approximate the incoming process at node i as a Poisson process with aggregate rate λ^i. Hence, in the same way as we used a tree of $M/M/\infty$ queues to model the network earlier, we can instead model the network as a tree of $M/M/k/k$ queues.

The $M/M/k/k$ formulation provides us with a means to adaptively design the buffering strategy at each node. If we suppose that the aggregate traffic levels arriving at a sensor node is λ, then the packet drop rate (the probability that a new packet finds all k buffer slots full) is given by the well-known Erlang Loss formula for $M/M/k/k$ queues:

$$E(\rho, k) = \frac{\rho^k}{k!} p_0 = \frac{\frac{\rho^k}{k!}}{\sum_{i=0}^{k} \frac{\rho^i}{i!}}, \tag{9}$$

where $\rho = \lambda/\mu$. For an incoming traffic rate λ, we may use the Erlang Loss formula to appropriately select μ so as to have a target packet drop rate α when using buffering to enhance privacy. This observation is powerful as it allows us adjust the buffer delay parameter μ at different locations in the sensor network, while maintaining a desired buffer performance. In particular, the expression for $E(\rho, k)$ implies that, as we approach the sink and the traffic rate λ increases, we must decrease the average delay time $1/\mu$ in order to maintain $E(\rho, k)$ at a target packet drop rate α.

5. RCAD: RATE-CONTROLLED ADAPTIVE DELAYING

A consequence of the results of the previous section is that nodes close to the sink will have high buffer demands and their buffers may be full when new packets arrive. In practice, we need to adjust the delay distribution as a function of the incoming traffic rate and the available buffer space.

In order to accomplish this adjustment, we propose *RCAD*, a Rate-Controlled Adaptive Delaying mechanism, to achieve privacy and desirable performance simultaneously. The main idea behind RCAD is buffer preemption—if the buffer is full, a node should select an appropriate buffered packet, called the *victim packet*, and transmit it immediately rather than drop packets. Consequently, preemption automatically adjusts the effective μ based on buffer state. In this paper, we have proposed the following buffer preemption policies:

—*Longest Delayed First (LDF).* In this policy, the victim packet is the packet that has stayed in the buffer the longest. By doing so, we can ensure that each packet is buffered for at least a short duration. The implementation of this policy requires that each node record the arrival time of every packet.

—*Longest Remaining Delay First (LRDF).* In this policy, the victim packet is the packet that has the longest remaining delay time. Preempting such packets can lessen the buffer load more than any other policy because such packets would have resided in the buffer the longest. The implementation of this scheme is straightforward because each node already keeps track of the remaining buffer time for every packet.

—*Shortest Delay Time First (SDTF).* In this policy, the victim packet is the one with the shortest delay time. By lessening an already short delay time, we expect that the overall performance will remain roughly the same. The implementation of this policy requires each node record the delay of every packet.

—*Shortest Remaining Delay First (SRDF).* In this policy, the victim packet is the packet that has the shortest remaining delay time. In this way, the resulting delay times for that node are the closest to the original distribution. As in the case of the LRDF policy, the implementation is straightforward.

6. EVALUATING RCAD USING SIMULATIONS

In this study, we have developed a detailed event-driven simulator to study the performance of RCAD. The simulations modeled realistic network/traffic settings, and measured important performance and privacy metrics.

6.1 Performance Metrics and Adversary Models

In our simulated sensor network, we have multiple source nodes that create packets, and intermediate nodes that follow RCAD schemes for buffering packets prior to forwarding them. As an important player of the game, the adversary stays at the sink, observes packet arrivals, and estimates the creation times of these packets.

In this study, we assume a powerful adversary that can acquire the following parameters for each flow: (1) the hop count of that flow, (2) the delay distributions for nodes along the flow, and (3) the traffic arrival process of the flow, for example, the arrival rate, the arrival distribution, etc. For an observed packet arrival time z, a baseline adversary estimates the creation time of this packet as $x' = z - y$, where y is the average delay of the flow, which the adversary can calculate from its knowledge of the delay distributions. In the simulations, we use the *square error* to quantify the estimation error, that is, $(x' - x)^2$ where x is the true creation time. Similarly, for a series of packet arrivals from the same flow $z_1, z_2, \ldots, z_m$, a baseline adversary estimates their creation times as $x'_1, x'_2, \ldots, x'_m$, and $x'_i = z_i - y$. The total estimation error for m packets is then calculated as the *mean square error*, $\sum(x'_i - x_i)^2/m$. We note that there is a direct relationship between mutual information and mean square error [Guo et al. 2005], and hence the scheme that has a higher estimation error consequently better preserves the temporal privacy of the source.

Since RCAD schemes dynamically adapt the delay processes by adopting buffer preemption strategies, it is inadequate for the adversary to estimate the actual delay times using the original delay distributions before preemption. As a result, we also enhance the baseline adversary to let the adversary adapt his estimation of the delays. We call such an adversary as an *adaptive* adversary.

In order to understand our adaptive adversary model, let us first look at a simple example. Let us assume there is only one node with one buffer slot between the source and sink. Further, assume that the packet arrival follows a Poisson process with rate λ, and the buffer generates a random delay time that follows an exponential distribution with mean $1/\mu$. If the buffer at the intermediate node is full when a new packet arrives, the currently buffered packet will be transmitted. In this example, if the traffic rate is low, say $\lambda < \mu$, then the packet delay time will be $1/\mu$. However, as the traffic increases, the average delay time will become $1/\lambda$ due to buffer preemptions. Following this example, our adaptive adversary should adopt a similar estimation strategy: at low traffic rates, he estimates the overall average delay y by h/μ, while at higher traffic rates, he estimates the overall average delay y as a function of the buffer space and the incoming rate, that is, hk/λ, where h is the flow hop count, k is the number of buffer slots at each node, and λ is the traffic rate of that flow. Given an aggregated traffic rate λ_{tot} from n sources converging at least one-hop prior to the sink, the adversary can compute the probability of buffer overflow via the Erlang Loss formula in Equation (9). He then can compare this against a chosen threshold and if the probability is less than the threshold, he will assume the average delay introduced by each hop is $1/\mu$. However, if the probability is higher than the threshold, the average delay at each node is calculated to be nk/λ_{tot}.

Additionally, we note that it is desirable to achieve privacy while maintaining tolerable end-to-end delivery latency for each packet. Hence, in our studies, for a network performance metric we use the average end-to-end delivery latency for packets coming from a particular flow versus the underlying traffic rate and the RCAD strategies employed.

6.2 Simulation Setup

The topology that we considered in our simulations is illustrated in Figure 2. Here, nodes S_1, S_2, S_3, and S_4 are source nodes and create packets that are destined for the sink. Thus, we had four flows, and these flows had hop counts 15, 22, 9, and 11 respectively. Each source generated a total of 1000 packets

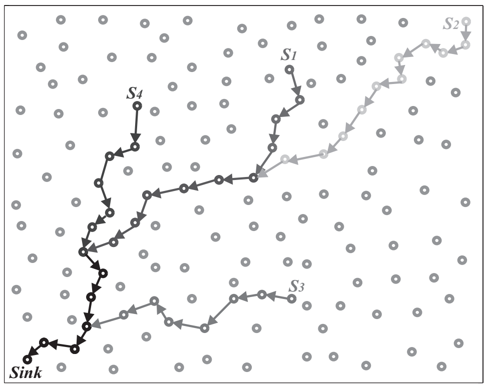


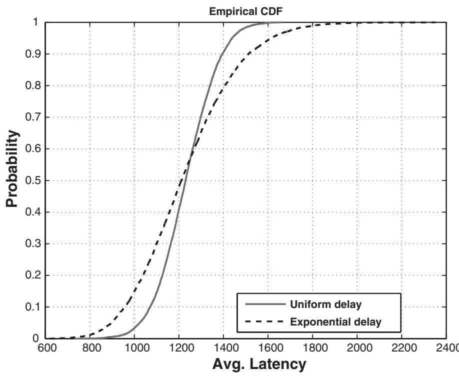
Fig. 2. Simulation topology.

with a mean interarrival time of $1/\lambda$ time units. In our experiments we varied $1/\lambda$ from 2 (i.e., the highest traffic rate) time units to 20 (the slowest traffic rate) to generate different cases of traffic loads for the network. The main focus of our simulator is the scale of the network, so we simplified the PHY- and MAC-level protocols by adopting a constant transmission delay (i.e., 1 time unit) from any node to its neighbors. When a packet arrives at an intermediate node, the intermediate node introduces a random delay following an exponential distribution with mean $1/\mu$. Unless mentioned otherwise we took $1/\mu = 30$ time units in the simulations. The results reported are for the flow S_1 to the sink.

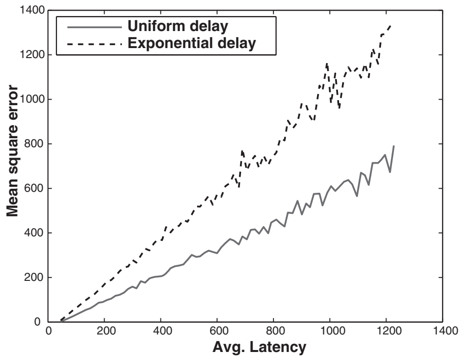
6.3 Performance Results

Before analyzing the performance of RCAD strategies, we illustrate how choosing exponential delay distribution achieves a better tradeoff between overall message latency and offering better uncertainty as compared to uniformly random delay distribution. Figure 3(a) shows the cdf of average end-to-end latency for both the distributions. As we can see they are pretty close to each other in terms of latency but the exponential delay distribution generates much larger error in an adversary's estimate of the time of origin of a message for the same average latency.

6.3.1 *Comparison of RCAD Strategies.* Figures 4(a) and (b) present the mean square error and the delivery latency of the four RCAD strategies where we assume each sensor node has a buffer of 10 slots (which is typical for a Mica2 mote), and a preemption-less strategy that assumes unlimited buffer space on each node. In this set of experiments, we used the baseline adversary model that estimated the delay for flow i as h_i/μ, where h_i is the hop count of flow i and $1/\mu$ is the average per hop delay (30 time units). At low traffic rates ($1/\lambda = 16, 18, 20$), these five strategies perform the same because the average buffer requirement per node is less than 10. As the traffic increases, the four preemption strategies lead to much higher mean square error, thus providing better temporal privacy. This is because the baseline adversary did not take into consideration the effect of buffer preemptions. Among

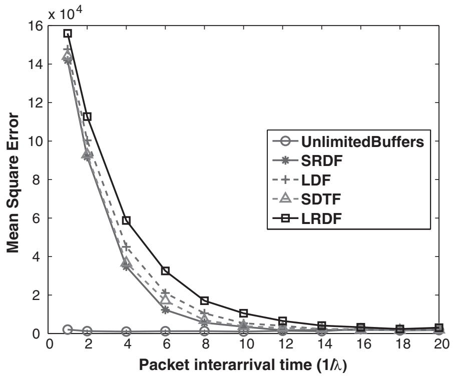


(a) Average Latency

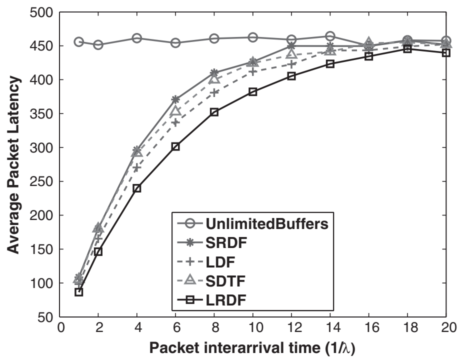


(b) Average Mean Square Error

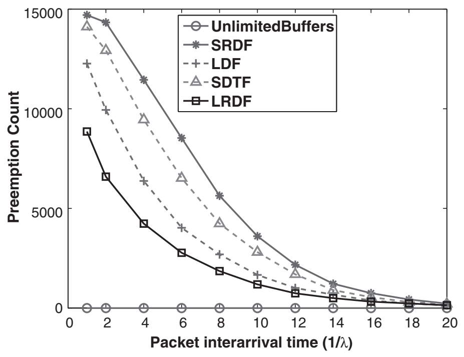
Fig. 3. Comparing expoential delay distribution to uniformly random delay distribution.



(a) Mean square error



(b) Delivery latency



(c) Number of preemptions

Fig. 4. Comparison of the four RCAD strategies and the scenario with unlimited buffers. We note the x-axis is in terms of average source interarrival time $1/\lambda$.

the four RCAD strategies, we observed that LRDF policy consistently performs the best in terms of privacy, followed by LDF and SDTF, while SRDF was the worst.

In order to understand the difference between these four strategies, let us look at more detailed statistics. Figure 4(c) presents the number of preemptions that occurred during the experiments. We observe the opposite order here: the strategy that provides the most privacy incurred the least number of preemptions. This may appear counterintuitive at first glance, but can be simply explained: the strategy that leads to more preemptions tends to alter the original delay distribution less, and thus confuses the adversary less. For example, LRDF selects the packet that has the longest remaining delay time as the victim packet. Preempting these packets will have two effects: (1) it will alter the original delay distribution more, and (2) it will reduce the number of preemptions.

Moving our attention to delivery latency, we observed that the preemptionless strategy with unlimited buffers incurred much longer latencies at higher traffic rates. Among the four preemption strategies, LRDF has the shortest latency because it tends to reduce the delay times in the buffer the most.

6.3.2 *Impact of Distance of a Source from the Adversary.* Figure 5 shows the impact distance has on the performance of RCAD algorithms. We compare the same metrics as above for four different sources as shown in Figure 2 and using LRDF. Note here that this comparison also allows us to answer the question of how RCAD would work if the adversary had compromised a fraction of the nodes in the network and programmed them never to delay any packets. We can think of this act as equivalent to the adversary reducing the number of hops between the source and the sink and hence get an idea of the impact of such a compromise on the temporal privacy using graphs similar to that shown in Figure 5.

6.3.3 *Reducing Preemption by Adopting Varying Delay Distributions.* Buffer preemption is necessary to avoid dropping packets due to buffer saturation, and we have just seen that it can help provide better temporal privacy. Buffer preemption, however, also has disadvantages, especially as it introduces additional protocol overhead at each sensor node associated with the selection of victim packets. Hence, in order to reduce protocol overhead we must reduce the frequency of preemption, but at the same time strive to maintain the same level of temporal privacy.

One strategy for reducing preemption is to let each node employ a different delay distribution. Since sensor networks usually have many more sources than sinks, as illustrated in Figure 1(c), nodes closer to the sink experience higher traffic volumes than nodes closer to the source. As a result, the nodes closer to the sink should delay packets much less in order to relieve the buffer requirements at these nodes. Our objective with this approach is to keep the buffer usage the same across all the nodes. As discussed in Section 4, the number of buffered packets at node i can be estimated as $\overline{N_i} = \rho_i = \frac{\lambda_i}{\mu_i}$. Suppose we consider a flow with h hops (i.e., h nodes before the sink), and use node 1 to

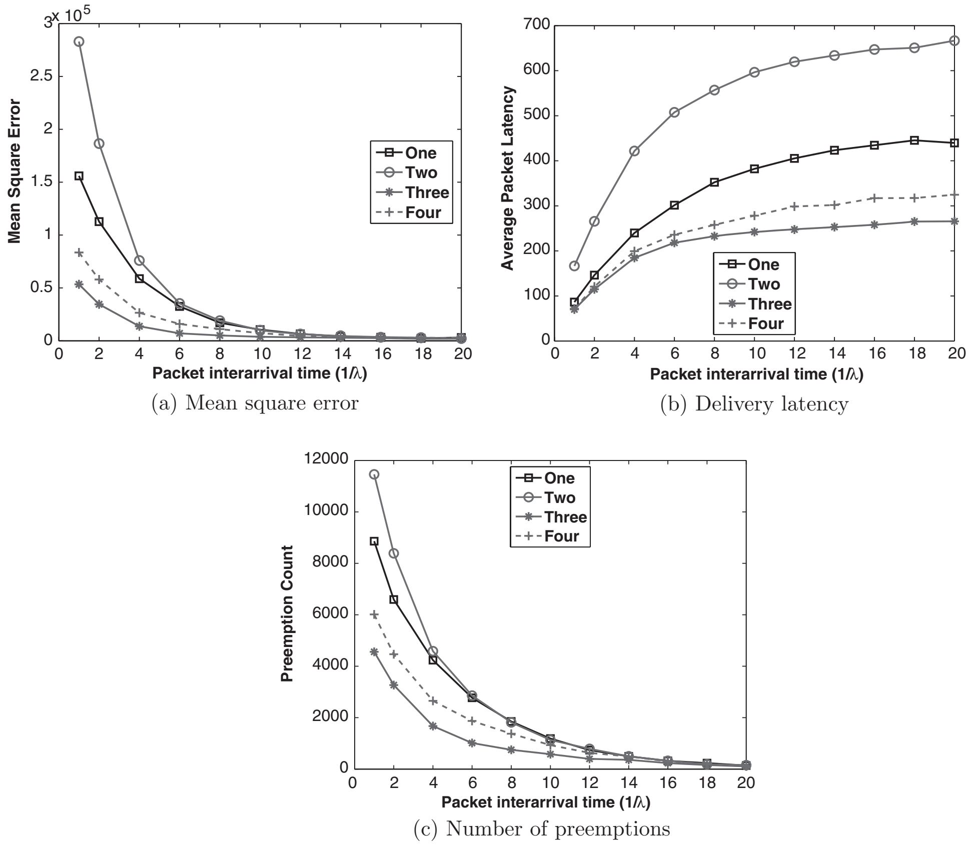
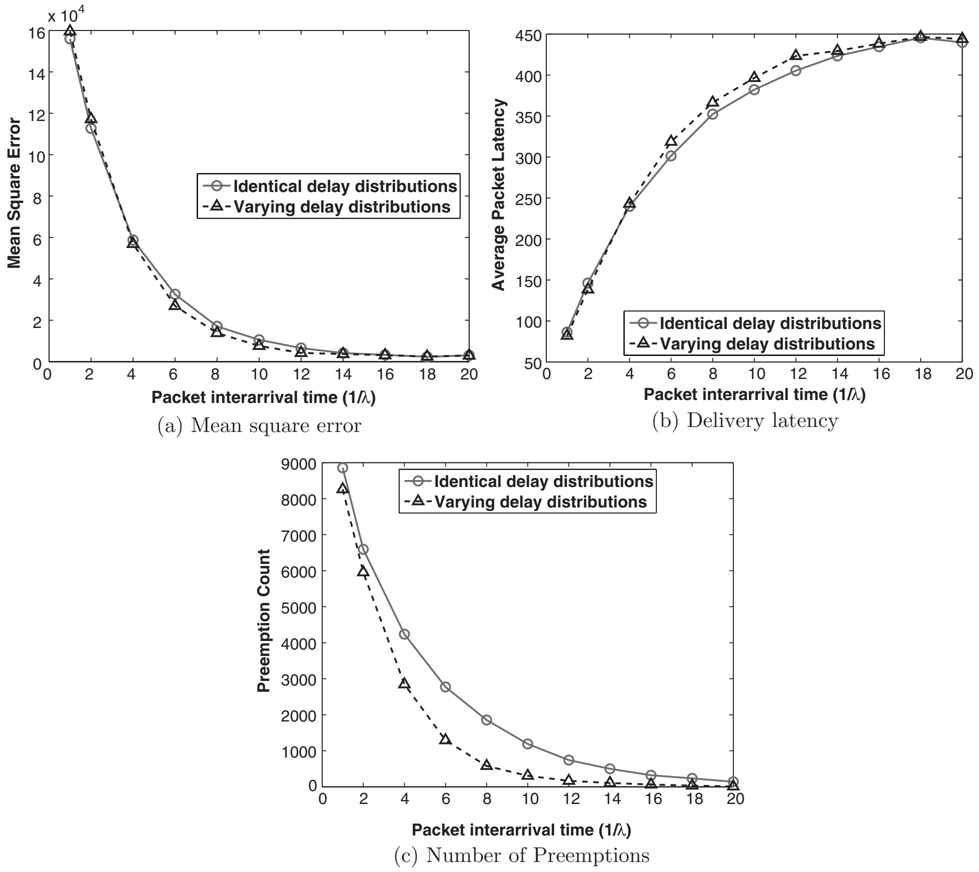


Fig. 5. Comparison of the LRDF algorithm for four different source-sink distances. We note the x-axis is in terms of average source interarrival time $1/\lambda$.

denote the last node before the sink and node h to denote the source node. To keep $\overline{N_i}$ constant across all nodes while having a target overall average delay of D, we choose the average delay time $1/\mu_i$ for node i as β/h_i, where β is the coefficient and h_i is the hop count between node i and the sink. Thus, we have $\sum_{i=1}^{h} \frac{\beta}{h_i} = \sum_{i=0}^{h-1} \frac{\beta}{i+1} = \beta(\gamma + \psi(h+1))$, where γ is the Euler-Mascheroni constant and $\psi(x)$ is the digamma function. Hence, the average delay time $1/\mu_i$ for node i is calculated as $\frac{D}{(i+1)(\gamma+\psi(h+1))}$.

We conducted a set of experiments to study the performance of RCAD strategies when using varying delay distributions chosen as above. The results with LRDF are presented in Figure 6. We observe that employing variable delays can significantly reduce the number of preemptions, especially for mid-range traffic rates. At the same time, having variable delays will not degrade either the mean square error or latency much. Although the preemptions were reduced by an amount up to 70%, the largest estimation error reduction we observed was 16%, while the largest latency increase was only 4%.



(a) Mean square error

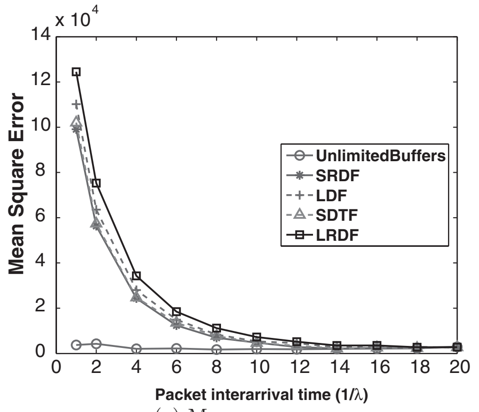
(b) Delivery latency

(c) Number of Preemptions

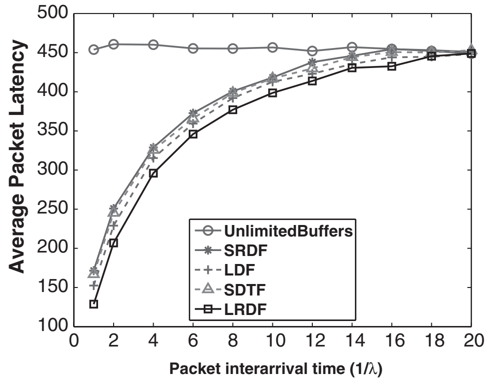
Fig. 6. Comparison of the performance of the LRDF policy when all nodes have identical delay distributions and when nodes have varying delay distributions.

6.3.4 *Impact of Aggregation.* Figure 7 demonstrates the RCAD performance in the presence of a single source of traffic and therefore no aggregation taking place at any node. We can see that the performance follows a trend similar to that in the presence of multiple sources with LRDF coming up the winner. Figure 8 shows head-to-head comparison of privacy protection offered to source in our topology in the presence or absence of other simultaneous sources of traffic. We can see that RCAD provides quite comparable performance even to a single stream.

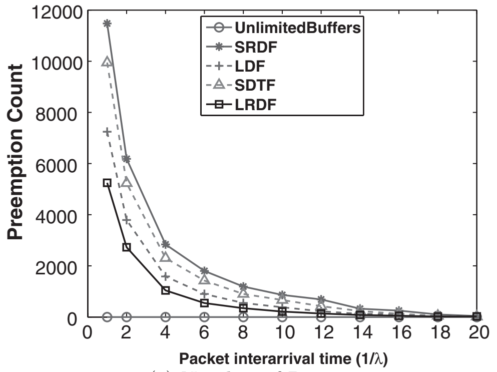
6.3.5 *Impact of Buffer Size.* Figure 9 shows the performance of RCAD algorithms with varying buffer sizes on the nodes. We can see that RCAD algorithms behave very well with low buffer sizes in fact better than when they have large buffers at their disposal. We hinted at this in earlier discussion but show concrete proof here. This happens because the smaller buffer sizes mean the RCAD algorithms deviate from the mean delay of their exponential delay distribution and thus the adversary is thrown off-base from his estimate which uses this mean delay.



(a) Mean square error



(b) Delivery latency

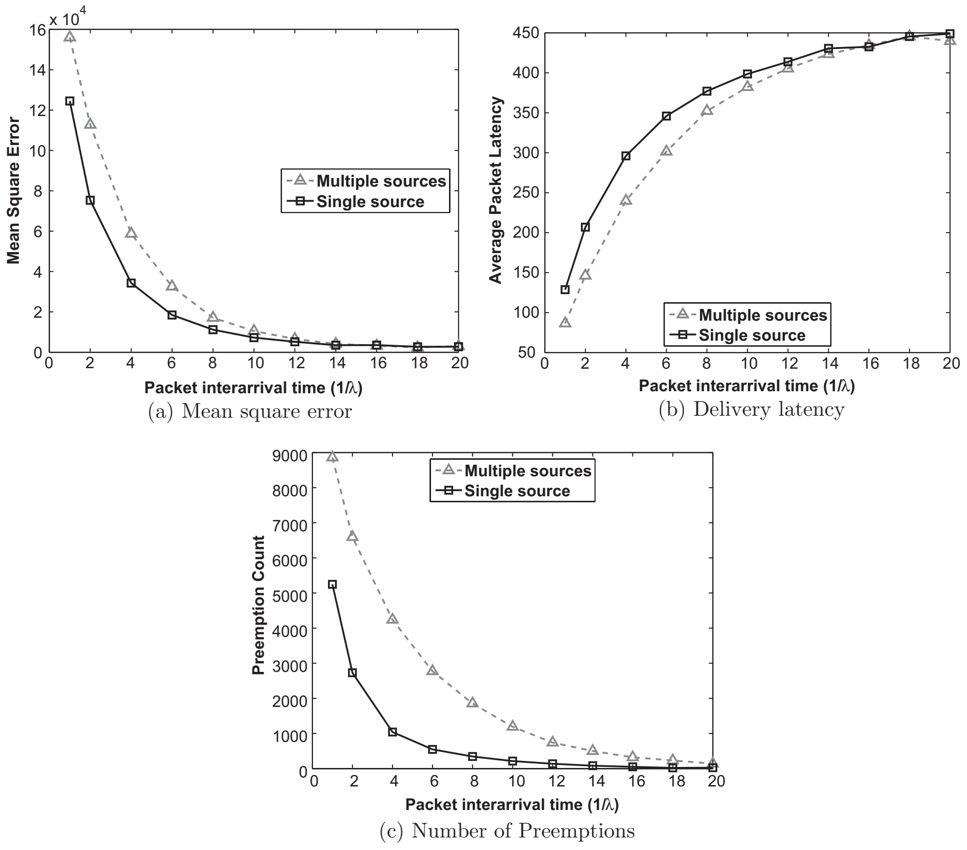


(c) Number of Preemptions

Fig. 7. RCAD Performance with a single source and no aggregation of traffic.

6.3.6 *The Adaptive Adversary Model.* A baseline adversary is inefficient in estimating delays for RCAD strategies with preemption. As a result, we studied the ability of an adaptive adversary to estimate the time of creation when using RCAD with identical delay distributions across the network. The resulting estimation mean square errors are presented in Figure 10. The adaptive adversary adopts the same estimation strategy as the baseline adversary at lower traffic rates, i.e. $1/\mu$ per hop, but it uses the incoming traffic rate to estimate the delay at higher traffic rates, that is, $h_i k/\lambda_i$ for the average delay of flow i. The switch between estimation strategies used the Erlang Loss formula for a threshold preemption rate of 0.1. Figure 10 shows that the adaptive adversary can significantly reduce the estimation errors, especially at higher traffic rates (lower interarrival times) where preemption is more likely.

An interesting observation is that at high traffic rates, the adaptive adversary can more accurately estimate the delay times generated by the LRDF policy when compared to other RCAD policies. Recall that earlier, LRDF had the highest estimation error against a baseline adversary while the SRDF policy had the least error (and hence the least privacy). Now, at high traffic rates against



(a) Mean square error (b) Delivery latency (c) Number of Preemptions

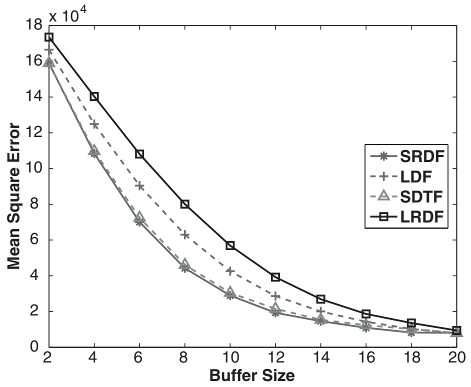
Fig. 8. Comparison of the performance of the LRDF policy for single and multiple sources.

an adaptive adversary, the trend is reversed—SRDF has the best privacy while LRDF has the worst.

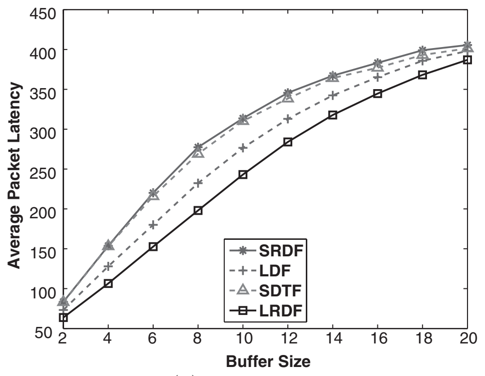
7. RELATED WORK

The problem of preserving privacy has been considered in the context of data mining and databases [Agrawal and Srikant 2000; Liew et al. 1985; Minsky 1976]. A common technique is to perturb the data and to reconstruct distributions at an aggregate level. A distribution reconstruction algorithm utilizing the Expectation Maximization (EM) algorithm is discussed in Agrawal and Aggarwal [2001], and the authors showed that this algorithm converges to the maximum likelihood estimate of the original distribution based on the perturbed data.

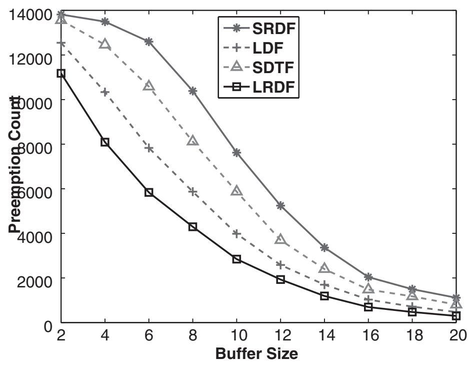
Contextual privacy issues have been examined in general networks, particularly through the methods of anonymous communications. Chaum proposed a model to provide anonymity against an adversary conducting traffic analysis [Chaum 1981]. His solution employs a series of intermediate systems called mixes. Each mix accepts fixed length messages from multiple sources and performs one or more transformations on them, before forwarding them in



(a) Mean square error



(b) Delivery latency



(c) Number of Preemptions

Fig. 9. RCAD performance with varying buffer sizes.

a random order. Most of the early mix related research was done on *pool mixes* [Diaz and Preneel 2004], which wait until a certain threshold number of packets arrive before taking any mixing action. Kesdogan et al. [1998] proposed a new type of mix, *SG-Mix*, which delays an individual incoming message according to an exponential distribution before forwarding them on. Later, Danezis [2004] proved using information theory that a SG-Mix is the optimal mix strategy that maximizes anonymity. The objectives of SG-Mixes, however, it to decorrelate the input-output traffic relationships at an individual node, and the methods employed do not extend to networks of queues.

Source location privacy problem in sensor networks is studied in Ozturk et al. [2004] and Kamat et al. [2005], where phantom routing, which uses a random walk before commencing with regular flooding/single-path routing, was proposed to protect the source location. Deng et al. [2004, 2005] proposed randomized routing algorithms and fake message injection to prevent an adversary from locating the network sink based on the observed traffic patterns.

Gruteser and Grunwald [2003] introduced a distributed anonymity algorithm that removes fine levels of detail that could compromise the privacy associated with user locations in location-oriented services. Hoh and Gruteser

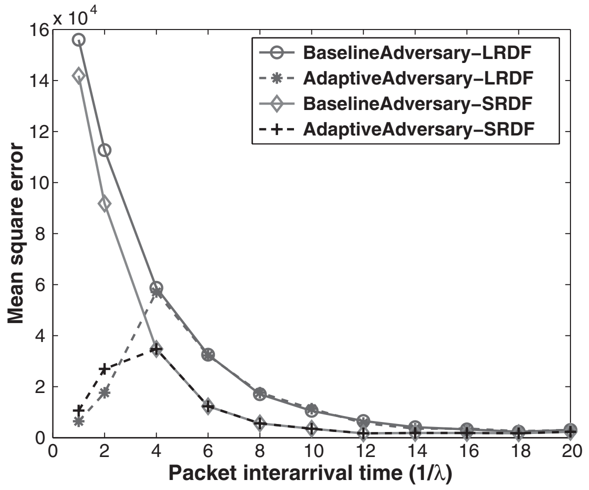


Fig. 10. The estimation error for the two adversary models, when LRDF and SRDF RCAD are employed.

[2006] proposed a path perturbation algorithm to reduce the probability of tracking users' path via trajectory-based linking continuously collected location samples.

8. CONCLUDING REMARKS

Preventing an adversary from learning the time at which a sensor reading was measured cannot be accomplished by merely using cryptographic security mechanisms. In this article, we have proposed a technique complimentary to conventional security techniques that involves the introduction of additional delay in the store-and-forward buffers within the sensor network. We formulated the objective of temporal privacy using an information-theoretic framework, and then examined the effect that additional delay has on buffer occupancy within the sensor network. Temporal privacy and buffer utilization were shown to be objectives that conflict, and to effectively manage the tradeoffs between these design objectives, we proposed an adaptive buffering algorithm, RCAD (Rate-Controlled Adaptive Delaying) that preemptively releases packets under buffer saturation. We then evaluated RCAD using an event-driven simulation study for a large-scale sensor network. We observed that, when compared with a baseline network consisting of unlimited buffers, RCAD was able to provide enhanced temporal privacy (the adversary had higher error in estimating packet creation times), while reducing the end-to-end delivery latency. Further, by adopting variable delays among the nodes along a routing path, we can reduce the number of buffer preemptions in RCAD without noticeably affecting privacy and network performance. Finally, we also devised an improved adversary model that can better estimate the delays produced by RCAD when compared to a naive adversary. In spite of the improved adversary model, RCAD is still able to protect the temporal privacy of sensor flows.

REFERENCES

AGRAWAL, D. AND AGGARWAL, C. C. 2001. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the Symposium on Principles of Database Systems*.

AGRAWAL, R. AND SRIKANT, R. 2000. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*. ACM Press, 439–450.

ANANTHARAM, V. AND VERDU, S. 1996. Bits through queues. *IEEE Trans. Inform. Theory 42*, 4–18.

BERTSEKAS, D. AND GALLAGER, R. 1992. *Data Networks*. Prentice Hall.

BOHGE, M. AND TRAPPE, W. 2003. An authentication framework for hierarchical ad hoc sensor networks. In *Proceedings of the ACM Workshop on Wireless Security*. 79–87.

CHAN, H. AND PERRIG, A. 2003. Security and Privacy in Sensor Networks. *IEEE Comput. 36*, 10, 103–105.

CHAUM, D. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Comm. ACM 24*, 84–88.

COVER, T. AND THOMAS, J. 1991. *Elements of Information Theory*. John Wiley and Sons.

DANEZIS, G. 2004. The traffic analysis of continuous-time mixes. In *Proceedings of the Workshop on Privacy Enhancing Technologies (PET'04)*, D. Martin and A. Serjantov, eds.

DENG, J., HAN, R., AND MISHRA, S. 2004. Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks (DSN)*.

DENG, J., HAN, R., AND MISHRA, S. 2005. Countermeasures against traffic analysis attacks in wireless sensor networks. In *Proceedings of the 1st IEEE/CreateNet Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*.

DIAZ, C. AND PRENEEL, B. 2004. Taxonomy of mixes and dummy traffic. In *Proceedings of the 3rd Working Conference on Privacy and Anonymity in Networked and Distributed Systems*.

GRUTESER, M. AND GRUNWALD, D. 2003. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

GUO, D., SHAMAI, S., AND VERDU, S. 2005. Mutual information and minimum mean-square error in gaussian channels. *IEEE Trans. Inform. Theory 51*, 4, 1261–1282.

HOH, B. AND GRUTESER, M. 2006. Protecting location privacy through path confusion. In *Proceedings of the 2nd International Conference on Security and Privacy in Communication Networks (SecureComm)*.

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)*.

KARLOF, C. AND WAGNER, D. 2003. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*. 113–127.

KESDOGAN, D., EGNER, J., AND BUSCHKES, R. 1998. Stop-and-go-mixes providing probabilistic anonymity in an open system. In *Proceedings of the 2nd International Workshop on Information Hiding*. Springer-Verlag, Berlin, Germany, 83–98.

LIEW, C. K., CHOI, U. J., AND LIEW, C. J. 1985. A data distortion by probability distribution. *ACM Trans. Datab. Syst. 10*, 3, 395–411.

MINSKY, N. 1976. Intentional resolution of privacy protection in database systems. *Commun. ACM 19*, 3, 148–159.

OZTURK, C., ZHANG, Y., AND TRAPPE, W. 2004. Source-location privacy in energy-constrained sensor network routing. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04)*.

PERRIG, A., SZEWCZYK, R., TYGAR, D., WEN, V., AND CULLER, D. 2002. SPINS: Security protocols for sensor networks. *Wirel. Netw. 8,* 5, 521–534.

SZEWCZYK, R., MAINWARING, A., POLASTRE, J., ANDERSON, J., AND CULLER, D. 2004. An analysis of a large scale habitat monitoring application. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*. ACM Press, 214–226.

Trappe, W. and Washington, L. 2002. *Introduction to Cryptography with Coding Theory*. Prentice Hall.

Tseng, V. and Lin, K. 2005. Mining temporal moving patterns in object tracking sensor networks. In *Proceedings of the International Workshop on Ubiquitous Data Management*.

Woo, A., Tong, T., and Culler, D. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03)*. 14–27.

Zhu, S., Setia, S., and Jajodia, S. 2003. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communication Security*. 62–72.

Received April 2008; accepted August 2008