RepEL: A Utility-preserving Privacy System for IoT-based Energy Meters

Phuthipong Bovornkeeratiroj, Srinivasan Iyengar†, Stephen Lee‡, David Irwin, and Prashant Shenoy
{phuthipong, shenoy}@cs.umass.edu, †t-sriyen@microsoft.com, ‡stephen.lee@pitt.edu, irwin@ecs.umass.edu
University of Massachusetts, Amherst; †Microsoft Research India; ‡University of Pittsburgh

***Abstract*—Modern Internet of Things (IoT) applications transmit sensor data to the cloud where it is subjected to analytics to provide useful services to users. Unfortunately, IoT sensor data often embeds sensitive private information that is vulnerable to leakage when sent to the cloud. Prior work on preserving IoT data privacy, particularly in the energy domain, focuses on obfuscating data to prevent extraction of private information. However, unless done carefully, data obfuscation significantly reduces the ability to extract useful but non-private information from the data. As a result, these existing techniques also reduce much of the utility derived from deploying IoT devices. In this paper, we address this problem by designing `RepEL`, a new utility-preserving privacy technique, which intelligently obfuscates smart energy meter data to prevent leaking a home's private occupancy information, while retaining the ability to perform useful energy disaggregation analytics.**

To preserve energy data's utility, our approach creates a randomized permutation of actual device usage via load replay while suppressing private user behavior information (such as occupancy) in the original data. We implement our algorithm on an embedded gateway node to demonstrate its feasibility and empirically evaluate our approach using real energy traces from homes. Our results show that the privacy leak rate for nearly two-thirds of the homes is below 10%, with four homes having no privacy leak. At the same time, the change in device usage for these homes is less than 3%. Further, we also demonstrate that `RepEL` has the flexibility to randomly replay loads, which can prevent adversaries from inferring behavioral patterns from device usage or use the information to determine occupancy.

I. Introduction

Recent advances in embedded systems hardware and wireless networking have led to the emergence of the Internet of Things (IoT) that is increasingly monitoring all aspects of our lives. Many consumer IoT products are now available for applications such as smart home automation, smart health, and others. Current IoT products use a cloud-based architecture, depicted in Figure 1, where the IoT device sends sensor data to the cloud, where it is subjected to analytics. Further, as shown in the figure, users often use a mobile app (or a voice assistant, such as Alexa) to interact with the cloud service and send commands to the IoT device via the cloud.

Such a cloud-based architecture for IoT devices, while commonplace, raises numerous privacy concerns. Since IoT sensor data often embeds private information about the user, sending this data to the cloud service implies the cloud analytics service now has the ability to extract this private information. In many cases, the embedded private information is *orthogonal* to the original purpose for which the data was collected by the IoT device, and can result in privacy leakage and unintended consequences. As an example, consider the privacy problems that arose when Strava, a popular fitness tracking app used by runners, published anonymized heat maps of popular running routes. Even though the data was anonymized by removing user identities, the location information embedded in the running routes inadvertently revealed the locations of secret military bases in remote regions [30]. Studies have shown that similar privacy leakages can occur from the use of smart home IoT products [19, 24, 25, 31, 32]. For example, when users operate their smart lights or smart appliances, the cloud service can infer a home's occupancy, i.e., whether it is occupied or vacant. Researchers [26] have shown that inferring occupancy is the first step towards launching various types of sophisticated privacy attacks (see Table I) such as when residents take vacations, when they eat out at restaurants, etc.

While the issue of IoT privacy has broad applicability to multiple application domains, in this paper, we focus on privacy in the domain of smart homes and energy. Maintaining user privacy in the face of cloud-based IoT services is a challenging problem. One privacy preserving approach is to use IoT devices "locally" without using any backend cloud services. However, this prevents the user from taking advantage of many useful features that cloud analytics services can provide. For example, such services use sophisticated machine learning to provide tailored recommendations for making the home more energy efficient or to identify anomalous energy usage based on comparisons across peer groups. Users can use these insights and recommendations to reduce energy waste and cut their energy bills.

Running such analyses locally is often infeasible due to the computational needs of these analytics algorithms or the inability of a standalone device to perform comparative analytics across users. Thus, not sharing IoT data with a cloud backend implies that the user has to forgo useful insights provided by these services. Another privacy-preserving approach is to use data obfuscation where the data is transformed by adding noise, or other means, prior to sending it to the cloud service [25, 36]. However, data obfuscation methods are a blunt instrument that, while enhancing privacy, destroy all useful information embedded in the data, whether private or not. As a result, running cloud analytics on this transformed data will produce erroneous results and is no longer useful to the user. Thus, traditional privacy-preserving techniques are often not compatible with cloud-based IoT services.

To address these drawbacks, we argue for a utility-

preserving privacy approach where the data produced by the IoT device is intelligently transformed to strip out private information while retaining other useful information embedded in the data. Such an approach will continue to allow cloud services to perform analytics that is useful to the user while preventing "mining" of private information. Figure 2 compares such as an approach to the current (non-private) cloud-based IoT services and recent privacy preserving methods for energy data. As shown, current IoT services preserve utility and sacrifice privacy; privacy-preserving data transformations based on obfuscations preserve user privacy but destroy utility. Our approach provides utility to users and respects privacy. While other recent efforts have examined utility-privacy tradeoffs [14, 16], this prior work has focused on theoretical issues, such as information theoretic analysis of this tradeoff or mathematically analyzing the impact of adding noise. Systems and algorithms for realizing these properties are still a nascent research topic and a key focus of our work.

We demonstrate the feasibility and efficacy of a systems approach to utility-preserving privacy by focusing on IoT-based energy meters that are common in smart homes. Examples include Sense, Engage, and others [33], all of which monitor the total electricity usage of the home at a fine granularity and transmit this data to the cloud for sophisticated energy analytics. Our approach can enhance the privacy of such energy devices by suppressing private occupancy information in the data while preserving the ability of their cloud services to perform advanced energy analytics.

Our utility-preserving privacy system for energy data, called `RepEL` (Replay Energy Loads), uses edge computing for its utility-preserving energy transformations. Specifically, it exploits edge nodes such as smart home gateways that are common in smart homes to interact with the energy meter for implementing its privacy data transformation algorithm. Our hypothesis is that by intelligently permuting the usage patterns of appliances and devices, our system can thwart timing-based occupancy attacks while retaining key appliance usage energy information necessary for performing cloud energy analytics. In designing, implementing and evaluating our `RepEL` approach, we make the following contributions.

- We present the problem of utility-preserving privacy in the energy domain by emphasizing occupancy-based privacy attacks and disaggregation-based energy analytics.
- We then present `RepEL`, a utility-preserving privacy algorithm, that transforms energy data in real-time to hide private occupancy information while retaining the ability to perform energy-efficiency analytics. Our technique uses an edge gateway node and an energy-storage battery to implement its record and replay privacy transformations. Further, our algorithm uses Metropolis-Hasting statistical sampling method to create a device usage schedule based on any user-specified distribution to mimic a behavioral usage pattern and reduce privacy leakage.
- Third, we evaluate `RepEL` using real energy traces and ground-truth occupancy and show that it can reduce privacy leakage from energy meters to below 10% for a majority of the homes, while limiting the change in device usage (i.e., reduction in utility) to less than 3%.

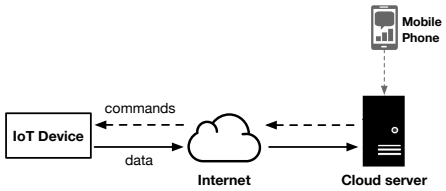


Fig. 1: Two-tier architecture of cloud-based IoT devices.

II. Background

In this section, we present background on IoT architectures and data privacy in the energy context.

A. Cloud-based IoT Devices

Today's IoT devices rely on a two-tier cloud-based architecture shown in Figure 3, where a device transmits data to the cloud over the Internet for analytics processing; the two-tier architecture also provides remote access to the device via the cloud. This is the basic architecture assumed in our work. Common IoT-based energy monitoring devices such as Sense and Engage are designed to track the total energy usage of a home at the main electric panel employ this architecture.

B. Occupancy-based Privacy attacks

Occupancy of a home or a building indicates when the building is occupied. The simplest type of occupancy is binary occupancy that simply indicates whether the home is occupied at any instant; occupancy can also be numeric and track how many occupants are present in the building. While binary occupancy has traditionally been tracked using motion sensors, studies have shown that such occupancy information can be easily determined from patterns of energy usage. This is because when a building is occupied, its occupants perform chores or activities such as cooking, laundry, watching TV, turning lights on or off, all of which manifest themselves in higher electricity usage or higher burstiness in observed usage. As a result, periods of occupancy and human activities become strongly correlated with periods of higher energy usage. This is depicted in Figure 3 which shows high electricity usage from foreground loads (e.g. water heater and cooking) in the mornings and evenings when residents are home; background loads (e.g. fridge) that always runs in the background does not reveal occupancy.

Researchers have previously studied these correlations between energy usage and occupancy [3, 13]. By statistically analyzing the energy data to detect periods of high burstiness or high usage, these efforts were able to infer occupancy periods with accuracies of 70 to 90% [8]. While occupancy information, by itself, seems innocuous, prior research [26] has shown that it is the first step towards launching more serious privacy attacks. Table I shows a list of sensitive information that is inferred once occupancy patterns of home are known [26]. As noted, private information such as when residents take

(a) Current IoT devices Obfuscation-based methods Our approach

Fig. 2: Privacy-utility comparison of various methods.

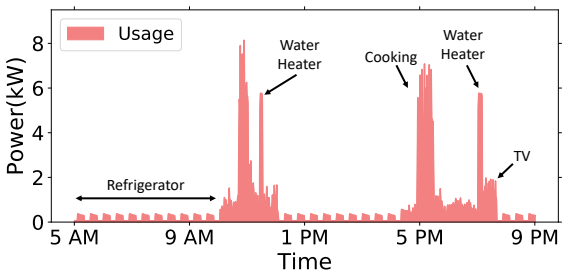


Fig. 3: Energy consumption data reveals periods of occupancy.

Private Question	Granularity needed
When do you take vacations?	Hourly
Do you eat out in the evenings?	Seconds
Were you home during your sick leave?	Hourly
Did you watch the game last night?	Seconds
Did you leave your child home alone?	Seconds
Did you get a good night sleep?	Seconds
Do you eat hot or cold breakfast?	Seconds

TABLE I: Private questions that can be learnt from occupancy patterns (see [26]).

vacations (inferred from long unoccupied periods), whether they eat out in the evenings (inferred from lack of occupancy and lack cooking activities in the evening), etc., can be learned once occupancy patterns are known.

C. Disaggregation-based Energy Analytics

Load disaggregation is a form of energy analytics that uses the total energy usage trace of a building, which is the sum of the energy usage from individual appliances and loads, and extracts ("disaggregates") the individual load usage from the total usage. The approach, also referred as *Non-Intrusive Load Monitoring (NILM)*, is possible since each load or device exhibits a unique energy signature and these patterns manifest themselves in the total usage when the device is active. This is shown in Figure 3 where usage patterns from the refrigerator, water heater, etc., are still visible in the aggregate energy usage trace. NILM methods exploit these unique energy signatures to discern those patterns and determine individual load usage from the total usage using methods ranging from step detection [15] to FHMM-based machine learning [5, 21, 22].

Disaggregation is usually the first step for performing more sophisticated energy analytics in the cloud. By itself, disaggregation reveals a breakdown of energy usage from various appliances (or groups) as shown in Figure 4. Once

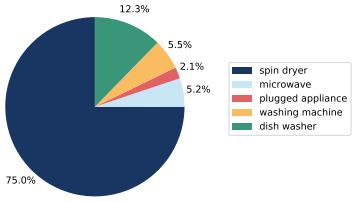


Fig. 4: Disaggregated usage for a sample home.

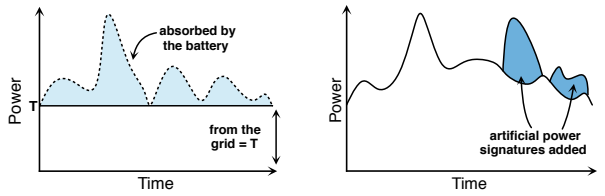
this information has been learned, cloud services can perform more advanced analytics such as anomaly detection to determine if the total usage of the home or the usage in any disaggregated category is higher than the norm. Anomalies can also be determined through comparative analytics [17] where the home's disaggregated usage is compared with other peer homes that are also reporting to the cloud to determine inefficient categories. Such analytics can even provide specific recommendations on what the homeowner should do to become more energy-efficient [17] and reduce energy bills.

Note that it is possible to perform load disaggregation even when the home has a net-metered solar installation. Net metering causes the electricity meter to record the difference between the total energy usage and the solar energy production, rather than the actual demand, which can confuse disaggregation algorithms. However, this is easily addressed by performing solar disaggregation[9], which uses weather-based models to separate out the actual energy load and the solar energy generation from the aggregated net-metered values. The separated output of solar disaggregation can then be fed to a load disaggregation algorithm to further compute a breakdown of the energy usage from individual loads.

D. Limitations of Preserving Energy Data Privacy

Prior work on privacy for energy data has not considered the issue of preserving utility in the data when transforming it for privacy. This body of work is based on data obfuscation, a method to alter the energy consumption recorded by the energy meter to prevent privacy attacks.

One approach, depicted in Figure 5(a) uses an energy storage battery "behind" the meter. Typically, when a device is in use, it draws power from the grid and this power draw is recorded by the energy meter. Some, or all, of the power draw of this device can be masked by serving it from the battery. In this case, since the device is effectively powered by the battery, it does not draw any additional power from the grid and its usage becomes invisible to the energy meter. Such a battery-based load smoothing approach, of which Non-



(a) Load Smoothing (b) Noise injection

Fig. 5: Ensuring data privacy through smoothing and noise injection as studied in [10, 36] remove data utility.

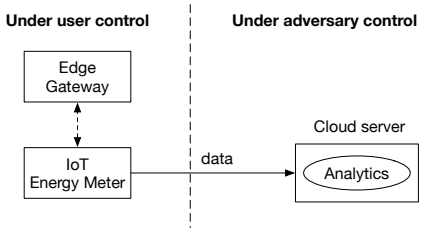


Fig. 6: Threat model for energy privacy.

Intrusive Load Leveling (NILL) and Lazy Step (LS) [25, 36] are two examples, smooth out the energy usage patterns seen by the meter through controlled charging and discharging of the battery. In doing so, the smoothened energy trace recorded by the meter no longer reveals high periods of energy use that are correlated to occupancy periods, but it also removes the ability to perform disaggregation-based analytics.

The second obfuscation approach, depicted in Figure 5(b), is based on noise injection [10] and involves randomly turning a large load on or off to create a noisy energy usage pattern at the energy meter; the approach in [10] employed an electric water heater to inject such noise. Since the random noise gets superimposed onto the actual usage, it prevents occupancy detection based on analyzing burstiness in the usage but also prevents load disaggregation. Table II compares existing privacy preserving methods with our approach.

E. Problem Statement

Given an IoT energy meter that records whole house energy usage, the goal of our work is to carefully transform the recorded usage such that cloud-based energy analytics can still be performed on the data, like before, while preventing occupancy-based attacks on this data.

III. RepEL: A Utility-preserving Privacy System

In this section, we present the threat model assumed in our work, followed by the design of our RepEL system.

A. Threat Model

The focus of our work is on IoT *data privacy and not security*. Thus, the issue of security of IoT systems where malicious adversaries steal data by hacking into the system, while a valid threat, is beyond the scope of our work.

Adversaries. The threat model assumed in our work is illustrated in Figure 6. We assume a homeowner who has deployed an IoT energy meter in their smart home. The smart home is assumed to contain a smarthome hub or a gateway node with limited computational capabilities. As shown, the IoT device and the gateway node are under the control of the homeowner and the homeowner can deploy any apps or code of their choice on the gateway (typically, a Raspberry PI-class node). The IoT energy meter streams recorded data to a cloud server; data is assumed to be encrypted during transmission. Once the data reaches the cloud, it is assumed to be under the control of the cloud service provider. The service provider is allowed to run any analytics of their choice on this data, and in our case, represents the adversary.[1]

As noted in Sec I, we assume it is impractical to run the energy meter as a local-only device without any cloud support as a possible privacy solution. This is because the types of advanced analytics that run in the cloud require significant computational resources—while it is feasible to perform simple types of disaggregation analytics locally on today's gateway-class nodes, more advanced machine learning analytics are beyond their computational capabilities. Further energy analytics that involves peer comparisons inherently need access to data from multiple devices, which can not be done locally. Consequently, our threat model assumes that the cloud provider is a necessary component in our system despite their ability to perform adversarial privacy attacks on the data.

Attack vectors. We assume that the cloud service provider will perform disaggregation analytics on the data prior to performing more advanced energy analytics. Doing so reveals both the energy usage from individual loads as well as the timing of when such loads were operated. The total energy usage pattern as well as the timing information about when various devices are used in the home then become attack vectors for learning when the home is occupied and then using occupancy for more sophisticated attacks (see Table I). While users may wish to reveal how much energy their loads are using for disaggregation-based energy analysis, they need to prevent the adversary from inferring the precise timing of when these appliances are being used (since doing so reveals private activity information as well as occupancy). Hence, the attack vectors considered in our work are (i) inferring binary occupancy information, and (ii) inferring the timings of when individual loads are operated within the household.

B. Overview

Next, we describe the intuition behind the RepEL approach. Our approach relies on a small energy storage battery in the home (e.g., Tesla Powerwall) and a gateway node that intelligently controls this battery (see Figure 7). This setup is identical to the battery-based load smoothing privacy methods in [25, 36]. Note that, our approach does require a battery in the home, similar to the privacy methods in [25, 36], and

[1]The service provider can also share this data with third parties, who then provide the analytics services, but this distinction is not important to our work.

Technique	Noise Injection [1]	Load Smoothing [10, 25]	RepEL
No Occupancy Detection	✓	✓	✓
No Activity Detection	✓	✓	✓
Billing	✓	✓	✓
Appliance Usage Period	✗	✗	✓
Building Efficiency Analysis	✗	✗	✓

TABLE II: A comparison of our approach with prior privacy methods for energy.

we believe that batteries, in conjunction with rooftop solar, will become commonplace in the coming decade. In some regions of the world such as India and parts of Africa that have an unreliable electric grid, batteries are already commonplace in residential homes to provide backup UPS power during blackouts. Their adoption in countries such as the USA is also growing since some states such as California are now requiring mandatory solar installations on all newly-built homes and providing generous subsidies for energy storage batteries to be installed as part of the solar system. As batteries become more commonplace, our approach becomes feasible using existing equipment with no additional investment.

The basic idea behind RepEL is to permute and randomize the order in which loads are seen by the energy meter while retaining all the original usage patterns exhibited by each individual load. Thus, each device appears in a randomly time-shifted and permuted order in the energy usage recorded by the energy meter. Since all device activities in the original data are still present in the permuted trace, disaggregation-based analytics continue to see the same information in the transformed data; however, since timings of device usage are randomized, timing-based occupancy attacks are no longer feasible. A consequence of permuting timing information of foreground loads is that any analytics that directly depends on timing information is no longer feasible on the transformed data; this tradeoff is necessary since timing information directly reveals occupancy and the only way to preserve private occupancy information is to supress fine-grain timing information through permutation.

This time-shifting and randomization is performed using the battery through a record and replay method. When a device is turned on by the user in the home, RepEL activates the battery, and all of the power draws of the device is serviced by the battery. The energy meter does not see the presence of the device since there is no grid power draw. Our system records the entire power usage pattern of the device during its active period (by measuring the current draw from the battery). It can then replay the recorded pattern at a later time by charging the battery intelligently so that the power drawn during charging mimics the power draw of the recorded device; since the battery charges from the grid, the meter "sees" the (mimicked) device at a later time. By recording and then replaying energy patterns of all foreground loads in a randomized fashion, our system enables energy analytics but hides the real timing information of when those loads were actually used in the home. Note that our system

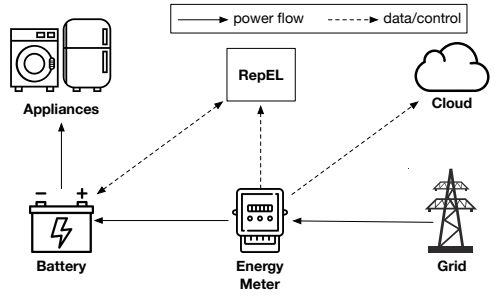


Fig. 7: Overview of our RepEL architecture.

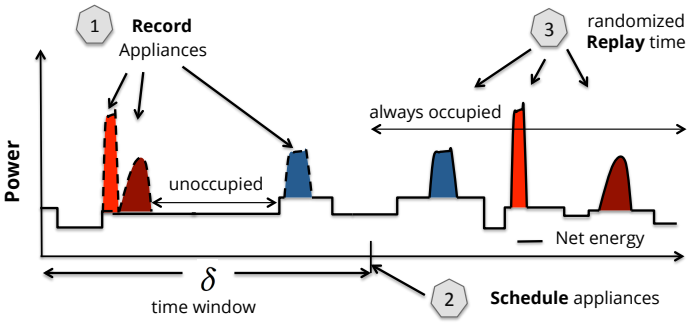


Fig. 8: Illustrates the three key steps — record, schedule, and replay — in the control algorithm.

leaves background loads unchanged since they does not reveal occupancy. In summary, our RepEL system modulates the battery such that it (i) discharges energy to mask foreground loads when they operate and (ii) charges the battery to replay foreground loads at a later time.

C. RepEL Control Algorithm

The objective of the control algorithm is to modulate the battery and preserve both utility and privacy. An empty battery cannot discharge energy to mask a load. Similarly, a full battery cannot charge to replay a recorded energy trace. Since battery sizes are limited, an intelligent control algorithm must balance the charge and discharge of the battery to preserve both privacy and utility. Figure 8 illustrates the three key steps in the RepEL control algorithm: *record*, *schedule*, and *replay*. In the **record** step, the operation of the foreground loads is registered and stored in a database over a user-defined time window. This forms the basis for replaying load at a later stage. The **schedule** step determines the operation

Key (Loads)	Value (Watt)
Dryer	[[2305, 1807,...],[2106, 1983,...]]
Stove	[[1233,1225,...]]
Microwave	[[803, 807,...],[798, 801,...],[800, 800, ...]]

TABLE III: Sample values in the key-value store for recording foreground loads.

schedule of recorded foreground loads to prevent adversaries from learning any behavior and preserve utility. Here, a user-specified probability distribution can be leveraged to generate a schedule for past loads. Finally, in the **replay** step, the battery's state of charge is used to mimic foreground loads to preserve privacy. Below, we describe each step in detail.

1) Record: `RepEL` records the energy consumption of a foreground appliance in a background process each time it is operated. We define foreground appliance as load that reveals human activity (e.g., load from a microwave). `RepEL` records energy traces as long as the battery continues to discharge energy to mask the load. The individual energy traces are captured using APIs exposed by energy meters or by running an online load identification algorithm [35]. These traces are stored in a *key-value* store, where the key represents the load type, and the values are a list of traces for different device runs. Table III shows a sample *key-value* store for a home over a period of δ = 1 day with a sampling interval of 1 minute.

2) Schedule: At the beginning of each interval δ, `RepEL` schedules the load recorded in the previous time window. A simple approach to schedule loads is to replay energy traces in random order. However, our approach allows the flexibility to replay the load from a user-defined distribution to mimic a phony *operational behavior*. In this general approach, if uniform distribution is selected, the scheduler will randomly replay the loads. However, if a custom distribution (say bimodal distribution) is selected, the operation time of the devices will closely mimic this distribution.

To enable a replay schedule using a target distribution P, we employ a Markov Chain Monte Carlo (MCMC) sampling method called *Metropolis-Hastings Algorithm* that can produce samples from any distribution. Since a target distribution may be difficult to sample from, MCMC approximates the distribution by generating a sequence of random sample values from a simpler distribution (e.g., uniform), referred to as the *proposal distribution*(Q), rejecting/accepting the samples with an acceptance ratio α. The acceptance ratio α indicates how probable is the candidate sample with respect to the target distribution. A higher acceptance ratio α means the sample is closer to the desired distribution and vice-versa.

Algorithm 1 presents the pseudo-code to generate a replay start time of loads from a desired target distribution. It takes as input a target distribution P, start times τ of K loads, and iteratively approximates the sample distribution. Specifically, it picks a candidate start time τ'_k of the k^{th} load from the proposal distribution and accepts with some probability α.

$$\tau'_k \sim Q(\tau'_k) = \text{Uniform}\{\tau'_{k-1}, \tau'_{k+1}\} \qquad \forall k \in K \qquad (1)$$

where $\tau'_0 = 0$ and $\tau'_{k+1} = \delta$, when $k = 1$ and $k = K$ respectively and represent the distribution's support values. The acceptance ratio α is computed using the target distribution to ensure that candidate samples from high density regions of the target distribution have a higher acceptance ratio. The algorithm returns a list of start times for load replay.

State of Charge	Action
Battery>=50%	Replay the smallest available load.
Battery<=50%	Replay the largest available load.
Battery==100%	Delay replaying the smallest available load. After replaying, call schedule step followed by the replay step.
Battery==0%	Immediately replay the largest available load. After replaying, call schedule step followed by the replay step.

TABLE IV: Replay action policies.

Algorithm 1 Determines start time schedule of loads

procedure GETSTARTTIMESCHEDULE(P, K, $iterations$)
  Initialize: $\tau \sim$ GetRecordedStartTime() // gets the recorded start time of loads from the key value store
  $old\_prob = \prod_{k=1}^{K} P(\tau_k)$
  for i in $[1..iterations]$ **do**
    for k in $[1...K]$ **do**
      $\tau'_k \sim \text{Uniform}\{\tau'_{k-1}, \tau'_{k+1}\}$
      $new\_prob = P(\tau_1) \cdot ... \cdot P(\tau'_k) \cdot ... \cdot P(\tau_K)$
      $\alpha = \frac{new\_prob}{old\_prob}$
      $u \sim \text{Uniform}\{0, 1\}$
      if $\alpha \geq u$ **then**
        $old\_prob = new\_prob$
        $\tau_k = \tau'_k$
  return τ

3) Replay: The algorithm uses the start time from the schedule step as an input to replay the foreground loads. Table IV describes our replay scheme that determines the *action* taken based on the battery's current state of charge. Based on this replay scheme, our replay algorithm is simple. The algorithm uses the schedule generated in the *schedule* step to replay a load from the key-value store. When the state of charge is 50%, our algorithm replays a trace from the key-value store that has the maximum energy footprint. This is done to aggressively discharge the battery and avoid running the risk of an empty battery. On the contrary, if the state of charge is greater than or equal to 50%, the algorithm replays a trace that has the minimum energy footprint. This will delay the battery from getting fully charged. After the load is replayed, the trace is removed from the key-value store.

If the battery is full, the algorithm cannot adhere to the start time schedule from the schedule step as load replay requires charging the battery. Thus, a replay of all load is

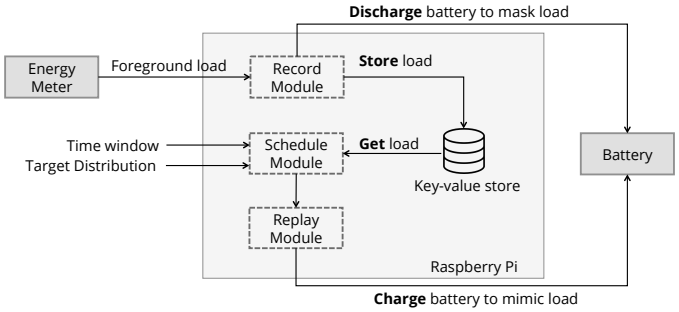


Fig. 9: Implementation details of the RepEL algorithm.

deferred and the schedule step is invoked to get new start times for scheduling the remaining loads. Further, we resume load replay when there is sufficient state of charge to replay the load with the smallest energy footprint. Note that the schedule step generates a start time for only those loads that have not been replayed. This is essential to ensure that the utility is preserved and prevent duplicate load profiles. Finally, if the battery is empty, we immediately replay a load with the largest energy footprint, ignoring the original schedule. This ensures that the battery is charged enough to mask foreground loads and prevent privacy leaks. Since the original replay schedule is not followed, the *schedule* step is invoked to get new start times for scheduling the remaining loads.

We note that our approach can be extended to a *vacation mode*, where the battery can replay loads from the previous week without recording any new ones until it is depleted or the owners come back. This will enable privacy of user even when the home is unoccupied. However, we note that to enable privacy for long periods(e.g., weeks) will require a larger battery — a known limitation in all privacy-preserving algorithms including ours.

IV. Implementation

Figure 9 depicts the implementation details of our RepEL system. We implemented RepEL's control algorithm in python on a Raspberry Pi. We assume energy meters can send aggregate energy to the cloud. Further, we assume that we cannot modify energy readings of smart meters after it is sent from the user's premises. Thus, our RepEL system is installed before the energy meters to provide local intervention. As shown, RepEL communicates with the energy meter to record the energy data of foreground loads in its key-value store. The foreground loads can be measured in two ways. In the first approach, RepEL can get the power consumption of individual foreground loads from advanced energy monitoring meters. Most energy meters such as eGauge can submeter individual loads by monitoring the circuits in the main electrical panel [12]. These energy meters measure power consumption at one-second resolution and can be accessed over an API.

In the second approach, we assume that such advanced energy meters are not available, and RepEL has access to only the aggregate energy profile of a home. In such cases, we can use non-intrusive load identification techniques [35] to identify and record the individual loads. Thus, foreground load from aggregate energy can be inferred and recorded in the key-value store. At the beginning of each time period δ, RepEL retrieves the recorded foreground loads from the key-value store and runs the *schedule* module to determine load replays. In the default setting, the schedule step assumes a uniform distribution to replay loads but any user-defined target distribution can be specified. Finally, the replay schedule and the energy loads are provided as an input to the *replay* module, which replays the energy load based on the scheme in Table IV. The control algorithm periodically tracks the battery's state of charge and modulates its charge/discharge rate. The replay module sends commands to the battery management system to charge at a certain rate, which is continuously modulated to mimic the foreground load's energy signature.

Characteristics	Dataset 1	Dataset 2
# of Homes	19	4 (summer)
Duration	1 Month (2016)	25-66 days (2012-13)
Occupancy	Not Available	Available
Total Appliances	100	16
# Appliances Runs	6021	1078
Granularity	1 Minute	1 Second
Location	Austin, TX	Switzerland

TABLE V: Key characteristics of the Dataport and the ECO dataset

V. Evaluation Methodology

A. Dataset Description

We use two datasets to evaluate our RepEL algorithm (see Table V). As discussed below, both datasets have fine-grained energy traces of appliances from different homes.

Dataset 1: Dataport from Pecan Street Inc. [11] The dataset contains plug-level energy traces at a one-minute resolution from residential buildings located in Austin, Texas. This data was collected for one month. For our evaluation, we selected 19 homes with at least two foreground appliances. Overall, there were 100 appliances in the buildings, which were operated a total of 6021 times by the occupants.

Dataset 2: ECO from ETH Zurich [7] The ECO dataset includes plug-level power traces of appliances along with ground truth occupancy data at a one-second resolution. This data was collected from six homes over a period of 4 months. However, our analysis uses data from four homes and only for periods where ground truth occupancy data was available. Overall, we found that occupancy information was available for a period ranging from 25 to 66 days. Further, the dataset consists of 16 appliances, which were operated 1078 times.

B. Experimental Setup

We use a trace-based simulation to evaluate our approach. However, we also study the feasibility of our approach on a gateway-class node such as a Raspberry Pi. For our evaluation, unless stated otherwise, we use a battery size of 13.5kWh[2],

[2]The capacity of a standard Tesla Powerwall batteries is 13.5 kWh.

and the default time period δ is set to one day to record and replay foreground loads. At the start of this time period, the foreground loads are scheduled and replayed by our approach. The foreground load schedule is determined by sampling the target distribution (set to be uniform) and uses 1500 MCMC iterations to sample. Finally, we use the Non-intrusive occupancy monitoring (NIOM) algorithm [8] to infer occupancy patterns and evaluate the efficacy of our approach in suppressing foreground loads.

C. Performance Metrics

To empirically evaluate our algorithm, we present two metrics that measure the privacy leakage of devices and the impact on the utility.

Privacy Leak Rate: We employ a more stringent metric to define privacy leak wherein partial suppression of the original foreground load is considered as leakage. This is because partial masked foreground load may be visible on the aggregate energy meter readings and reveal private information. Thus, if the foreground appliance is running, but the battery does not have sufficient energy to mask the load, then we consider it to leak private information. We evaluate the efficacy of our algorithm using the *privacy leak rate* defined as the percentage of foreground load operations that are not entirely masked by the energy storage. A value of 0% represents complete privacy, whereas 100% denotes that the battery was unable to suppress any foreground load.

$$privacy\_leak\_rate = 100 \times \sum_{i=1}^{N} \frac{is\_leak(i)}{N} \tag{2}$$

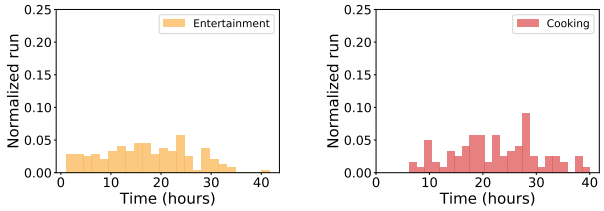
$$is\_leak(i) = \begin{cases} 1, & \text{if } i^{th} \text{ run of the appliance} \\ & \textit{and} \text{ battery is empty} \\ 0, & \text{otherwise} \end{cases}$$

where N is the total number of runs of all appliances for a given period.

Device Usage Change: `RepEL`'s algorithm alters the energy profile of foreground loads to suppress its footprint in the aggregate load. However, it replays this load at a later period to preserve the utility aspect. This maintains the device usage but obfuscates *when* the device was operated, preventing adversaries from inferring user behavior. The aggregate energy use of appliances can provide insights such as energy-efficiency of the device. To analyze the difference between the original and the replay profile of an appliance, we introduce *device usage change (duc)* that captures the error in replaying energy profile of loads and defined as.

$$duc = 100 \times \frac{\sum_{i=1}^{N}(replay\_profile_i - energy\_profile_i)}{\sum_{i=1}^{N} energy\_profile_i} \tag{3}$$

where N is the total number of runs of all appliances for a given period, $energy\_profile_i$ represents the load of the i^{th} run, and $replay\_profile_i$ represents the replay of the energy load of the i^{th} run. A value close to 0% indicates that the replay schedule is completely mimiced all the foreground loads, whereas 100% means the algorithm didn't replay any load.



(a) (b)

Fig. 10: Replay time distribution of loads across for a sample home. The loads are shifted from the original start time and help suppress user behavior information such as occupancy.

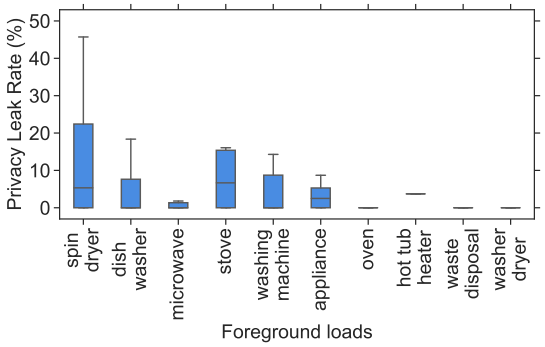


Fig. 11: Privacy leakage distribution of appliances across all homes using a 13.5kWh battery.

VI. Results

In this section, we summarize the results of exhaustively evaluating the performance of our algorithm.

A. Impact on privacy leakage

Figure 10 shows the replay time distribution of two loads for a sample home. A value close to zero indicates that the loads were replayed at the same time as the original time. As seen in the figure, the foreground loads linked to user behaviors are shifted from the original time and ranges from 1 hour to 41.6 hours. Since we use a uniform distribution to schedule replays, the loads are randomly permuted and replayed throughout the day. This can thwart any timing-based attacks that infer when devices are operated and related human behavior. On average, the replay time of a load is shifted by 19 hours.

We now examine the efficacy of our algorithm in preventing privacy leakage across all homes. Note that our metric penalizes us even when the battery fails to mask a small segment of the load profile. Thus, the privacy leak metric is strict and looks at the worst-case scenario. We ran `RepEL` to mask private information across all homes in the Dataport dataset and observed that 4 homes had no privacy leak. Further, nearly two-thirds of the homes have a privacy leak rate below 10%.

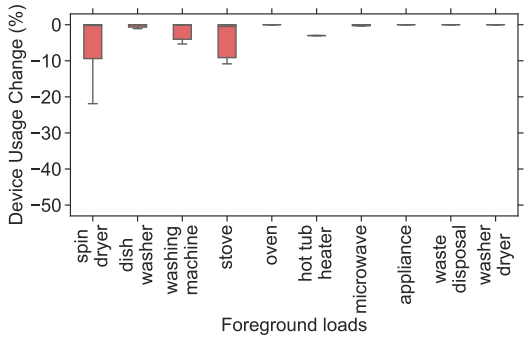


Fig. 12: Device usage change distribution of appliances across all homes using a 13.5kWh battery.

We also observe that homes that have higher privacy leakage have large loads that are operated for longer durations. Figure 11 shows the privacy leak rate per-appliance across all homes. As seen in Figure 11, smaller loads such as ovens have a *zero* privacy leak rate. On the other hand, larger loads such as spin dryers tend to consume more energy. In our evaluation, we used a 13.5 kWh battery. While this is sufficient to mask smaller loads, it may not be as effective in masking appliances with moderately high energy footprint. In particular, we observe that even for a spin dryer, a large load, the median privacy leak rate is 5.3%. For smaller loads, the median privacy leak rate is much less. We also evaluated the privacy leakage using a battery size of 27 kWh (not shown in the figure). We found that the median privacy leak rate of a spin dryer is 0.69% — a reduction of 86.9%.

Result: Transformations made by `RepEL` *preserve privacy by having a very low privacy leak rate of less than 10%.*

B. Impact on device usage

We measure the utility by determining the change in the device usage due to record and replay of foreground loads. It is important to note that our approach only replays a load when there is sufficient battery capacity available. And, the replay is *non-preemptive*, that is, the battery decides to replay the entire load only when it has sufficient capacity. This ensures that the energy data, and thus its utility, is maintained for any energy-efficiency analysis. We ran `RepEL` on all the homes in the Dataport dataset and found that more than *two-thirds* of them have device usage change that is less than 3%. Seven homes had perfect accuracy and no change in device usage.

We next analyze the appliances that impact device usage change. Figure 12 shows the device usage change per-appliance across all the homes. We observe that homes with large loads tend to have higher change in device usage. As shown in the figure, the median device usage change is nearly zero for almost all of the foreground loads. For spin dryers, one of the large loads, the average change in device usage is 6.91%. We also evaluated the change in device usage using a bigger battery size of 27 kWh. We found that spin dryers can preserve up to 98.1% of its original usage.

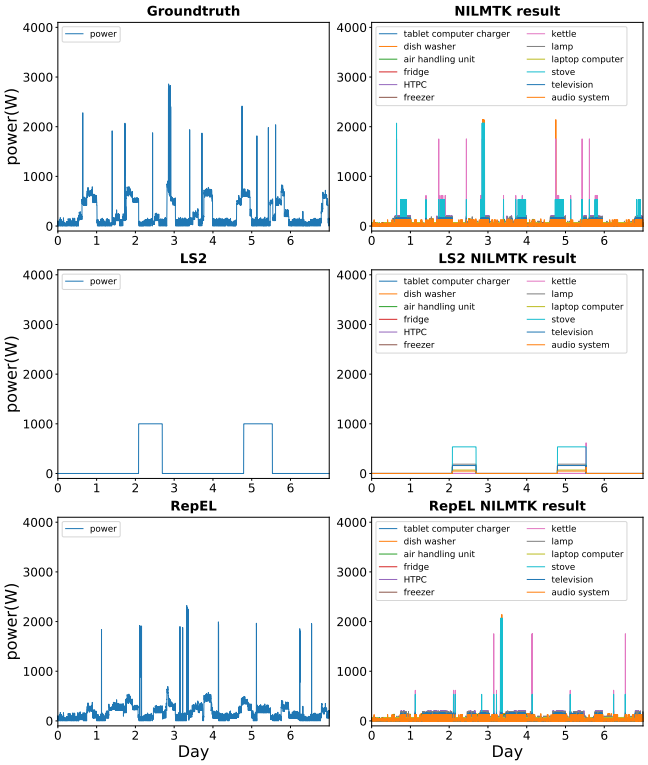


Fig. 13: Figures on the left column depict the final energy consumption of a home as seen by third-parties. The right column shows the disaggregated energy trace determined using a NILM algorithm (Combinatorial Optimization[5]). (a) Original Trace. (b) Lazy Stepping (c) RepEL.

Metric	LS2	RepEL
MAPE	64.25	6.53
MSE	4.07e+11	1.36e+10
Relative Error	0.26	0.06

TABLE VI: Disaggregation accuracy of `RepEL` and L2.

Result: Tranformations made by `RepEL` *maintain utility with less than 3% error in device usage from the ground truth.*

C. Preserving load disaggregation analytics

We ran the combinatorial optimization algorithm[3], a NILM technique, to perform energy disaggregation on the output from different privacy preserving techniques. Figure 13 depicts the energy consumption of a home as seen by third-parties using `RepEL` and Lazy stepping algorithm (LS2) [36]. As shown in the figure, LS2 changes the original trace to "flat" steps, thereby including very little actual usage information. However, `RepEL` algorithm maintains the likeness of the original trace but randomizes the order. The plots on the right show the disaggregated output of NILM where our NILM output resembles the original. This is quantified in Table VI

[3]The combinatorial optimization technique identifies a set of appliances that reconstructs the original trace[5].

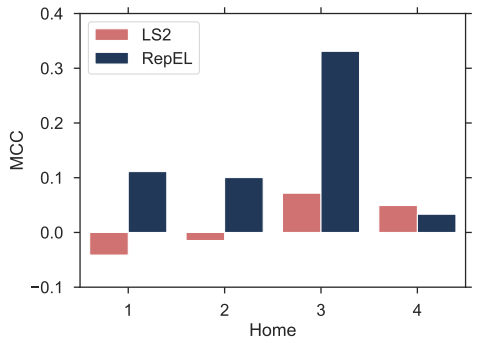


Fig. 14: Comparison of an occupancy detection algorithm's performance on the original trace with our approach.

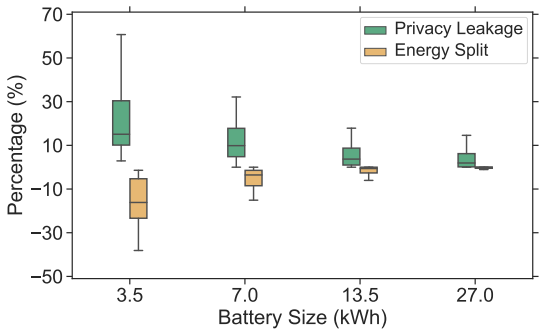


Fig. 15: Effectiveness of our algorithm in preserving privacy and utility for different battery sizes.

which shows lower errors for RepEL; MAPE and mean square errors for LS2 are an order of magnitude higher;
Result: RepEL enables energy disaggregation with high accuracy on transformed data, while LS2 yields high errors.

D. Preventing occupancy detection

We now show the effectiveness of our algorithm in masking occupancy patterns. To do so, we ran a state-of-the-art occupancy detection algorithm [8] on RepEL and LS2 outputs to determine occupancy for a given home. We use the Matthews Correlation Coefficient (MCC) to compare the quality of detection as it provides a balanced measure even when there is an imbalance in the occupancy patterns, i.e., the case where the homes are less occupied or always occupied. A correlation coefficient of +1 indicates that the observed and predicted match perfectly, whereas 0 shows that the predictions are random, and -1 represents total disagreement.

We use the ECO dataset for our experiment since it contains ground truth occupancy for each home. Figure 14 shows the MCC values for RepEL and LS2 for each of the 4 homes. As can be seen, both approaches yield low MCC values, which indicates good efficacy for masking occupancy. LS2 yields MCC values of -0.04 to 0.07, while RepEL yields MCC value of 0.03 to 0.11 for homes 1, 2 and 4. Note that RepEL has a higher MCC value of 0.33 on Home 3 since it represents the worst case for our approach by virtue of being occupied for long periods—this limits the ability of RepEL to exploit unoccupied periods to replay loads. Even so, this worst case scenario still yields a relatively low MCC of 0.33. LS2, in contrast, flattens the usage to mask occupied period and yields a lower MCC. Overall, RepEL sacrifices a small amount of privacy relative to LS2 in exchange for large gains in data utility. Figure 13 shows that LS2 nearly entirely eliminates any utility in the data, as LS2's disaggregation output is not reflective of the ground truth data.
Result: RepEL is effective at preventing occupancy detection.

E. Impact of varying battery sizes

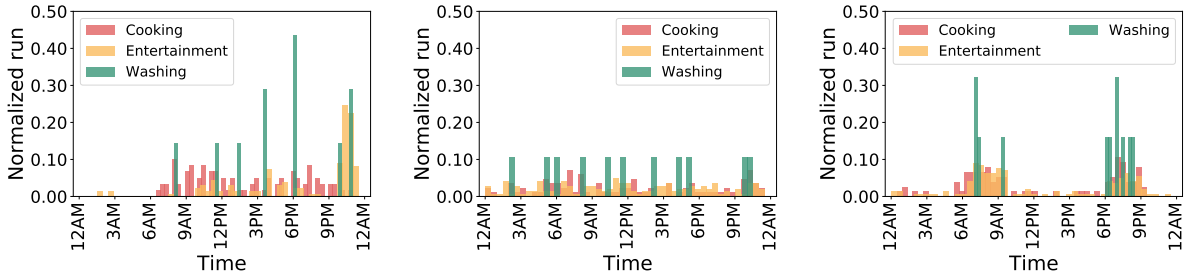
We now analyze the impact of battery sizes on preserving both privacy and utility of different appliance types. Figure 15 shows the range of privacy leak rates across all homes with varying battery sizes from the Dataport dataset. As shown in the figure, an increase in battery size reduces privacy leakage. This is because larger battery sizes can provide energy for longer periods, and thus, masks foreground loads. In particular, we observe that using a battery size of 27 kWh, the privacy leak rate of the median home is 1.92%.

Figure 15 shows the change in device usage across all the homes with varying battery sizes. The graph shows that the change in device usage reduces with an increase in battery size. This is because a larger battery size can replay foreground loads for a longer duration, and thus, mimic more foreground loads. In particular, we observe that using a battery size of 27 kWh, the change in device usage of the median home is nearly zero, indicating that we can entirely preserve the utility.

F. Preserving privacy via custom target distribution

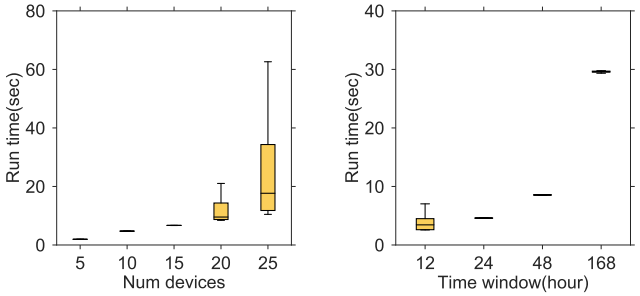
RepEL's algorithm allows a user to define a custom target distribution for scheduling loads, which provides the flexibility to replay a load at any specific time and mimic a particular activity. We illustrate this flexibility by replaying different runs of the foreground loads from a house in the Dataport for the entire duration. Figure 16(a) shows the normalized runs of foreground loads using the original trace of a home. We observe that we can infer behavioral patterns from the graph, such as the occupants usually watches television in the evening, depicted by increased usage between 10 pm to 11 pm. Similarly, cooking activity shows two peak periods — a morning peak and an evening peak.

Figure 16(b) shows the replayed schedule of the foreground loads based on a uniform distribution for the entire duration. As shown in the figure, our algorithm randomly scrambles the actual device usage period and replays load based on the target distribution. Since our target distribution is set to uniform, all the appliances are scheduled to replay uniformly. Thus, no adversary can determine a behavioral pattern device usage or use the information to determine occupancy. Similarly, Figure 16(c) shows the replayed schedule for the same set of power traces based on a bimodal distribution as target distribution. As shown in the figure, the replay of the appliances is two mimic two peak periods — a morning and



(a) Original distribution (b) Uniform distribution (c) Bimodal distribution

Fig. 16: `RepEL` replays foreground loads based on any target distribution to preserve privacy.



(a) Num of devices (b) Time window δ

Fig. 17: Figure (a) and (b) shows the execution time of MCMC sampling with varying input parameters on a Raspberry Pi.

an evening peak.

Result: `RepEL` has the broad ability to mimic any target distribution for scheduling and replaying loads.

G. Feasibility study on a Raspberry Pi

We analyze the execution time of our algorithm on a Raspberry Pi 3 and also study the feasibility of running disaggregation analytics locally on a gateway class node. Note that *record* and *replay* steps in our algorithm are executed in real-time. Hence, we only analyze the run time of the *schedule* step in our algorithm for varying time windows τ and the number of foreground loads.

We first set the time window δ to one day and run our simulation multiple times to determine the variability in execution time. Figure 17(a) shows the run time of the schedule step for an increasing number of foreground loads. As shown in the figure, the execution time increases with an increase in the number of devices. This is because the schedule step entails sampling the target distribution for each appliance to determine the possible start times for replaying the load. In particular, we observe that with five foreground devices, our algorithm takes less than 2 seconds to learn the start time for replaying the recorded appliances. With 25 foreground appliances, it takes less than a minute to compute the replay start times. Thus, it is feasible to run the sample step with almost no interruption to the record and replay steps.

Figure 17(b) shows the run time of the schedule step for varying time windows on a single home with ten foreground loads. We observe that the schedule step less than 5 seconds with time periods of one day or less. Even when the time window is set to a week, the schedule step takes less 32 seconds. Note that the schedule step runs at the start of the time window and can run in the background. Hence, our algorithm can be easily implemented using off-the-shelf components.

VII. Related Work

Recent studies have shown that energy usage from smart meters can reveal a significant amount of private information [5, 15, 29, 34]. As recent work demonstrates, energy consumption from a home can leak a person's daily activities [34], home occupancy patterns [3, 20, 23], and types of appliances at home [5, 15, 28, 29]. Unlike traditional privacy attacks, these *non-intrusive* techniques do not require additional sensor information to leak private information. For instance, studies have shown that simple statistical metrics of aggregate power usage can help predict when a home is occupied [20]. Similarly, operating an interactive load such as a television usually indicate occupancy [8]. Prior studies have also shown how non-intrusive techniques that can disaggregate energy to identify power contributed by individual appliances. Such techniques typically use hidden Markov models to identify different appliance signatures [6].

Mitigating such privacy attacks have been studied before [4, 10, 18, 25, 27, 36]. Prior studies have mostly focused on obfuscating the energy signal by adding noise to mask the variance in the load and provide privacy guarantees [25]. In [1], the authors proposed a differential privacy model by adding Laplacian noise to the data. However, the addition of such noise in the data does not preserve the device signatures in the signal, and does not thwart occupancy detection. Noise can be easily added in the energy signal by charging or discharging energy storage or through the use of loads such as water heaters [10, 25, 32]. Such control of energy storage can smooth out the load to prevent leakage of information. However, such techniques that smooth out the variations do not preserve the utility of the data. In contrast, our work suppresses any information that may cause occupancy leakage while preserving the utility of the data. Finally, drawing from concepts

in differential privacy, there has been work to show that power consumption of televisions can be made differentially private using batteries [2, 4]. However, unlike our approach, these techniques do not prevent occupancy attacks or attempt to preserve the device usage signature, which can be used to determine the usage of the appliances.

VIII. Conclusions

In this paper, we considered the privacy of energy data from smart meters and argued that while existing techniques enhance privacy, they do not preserve utility as they destroy all useful information embedded in the data, whether private or not. We presented the notion of utility-preserving privacy, which prevents privacy leakage while retaining the ability to perform energy-efficiency analytics. We designed and implemented `RepEL`, a utility-preserving privacy approach, that maintains appliance usage information while thwarting privacy attacks by randomly replaying energy loads to suppress private information leaks. We implemented our algorithm on a Raspberry Pi to demonstrate its feasibility and used real energy traces to analyze its performance. Our results showed that the privacy leak rate for nearly two-thirds of the homes is below 10%, with four homes having no privacy leak. At the same time, the change in device usage for these homes is less than 3%. Further, we demonstrate that `RepEL` has the flexibility to randomly replay loads, which prevents adversaries from inferring behavioral patterns from device usage.

Acknowledgment

We thank our shepherd Kui Ren for guidance and the anonymous reviewers for their insightful comments. This research is supported by NSF grants 1505422, 1763834, 1645952 and ARL contract W911NF-17-2-0196.

References

[1] G. Ács and C. Castelluccia. I Have a DREAM! (DiffeRentially privatE smArt Metering). In *International Workshop on Information Hiding*, 2011.

[2] G. Acs, C. Castelluccia, and W. Lecat. Protecting against physical resource monitoring. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, 2011.

[3] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, and T. Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, 2010.

[4] M. Backes and S. Meiser. Differentially private smart metering with battery recharging. In *Data Privacy Management and Autonomous Spontaneous Security*. 2014.

[5] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh, and M. Srivastava. Nilmtk: an open source toolkit for non-intrusive load monitoring. In *Proceedings of the 5th international conference on Future energy systems*, 2014.

[6] N. Batra, A. Singh, and K. Whitehouse. Neighbourhood nilm: A big-data approach to household energy disaggregation. *arXiv preprint arXiv:1511.02900*, 2015.

[7] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, and S. Santini. The eco data set and the performance of non-intrusive load monitoring algorithms. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, 2014.

[8] D. Chen, S. Barker, A. Subbaswamy, D. Irwin, and P. Shenoy. Non-intrusive occupancy monitoring using smart meters. In *Proceedings of the Workshop on Embedded Systems For Energy-Efficient Buildings*, 2013.

[9] D. Chen and D. Irwin. Sundance: Black-box behind-the-meter solar disaggregatio. In *the Eighth International Conference on Future Energy Systems*, May 2017.

[10] D. Chen, D. Irwin, P. Shenoy, J. Albrecht, et al. Combined heat and privacy: Preventing occupancy detection from smart meters. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2014.

[11] Dataport. pecan street., 2018. http://www.pecanstreet.org/.

[12] egauge energy monitoring solutions, 2018. http://www.egauge.net/.

[13] V. Garg and N. Bansal. Smart occupancy sensors to reduce energy consumption. *Energy and Buildings*, 32(1), June 2000.

[14] Z. Guan, G. Si, J. Wu, L. Zhu, Z. Zhang, and Y. Ma. Utility-privacy tradeoff based on random data obfuscation in internet of energy. *IEEE Access*, 5, 2017.

[15] G. W. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 1992.

[16] M. Hattori, T. Hirano, N. Matsuda, R. Shimizu, and Y. Wang. Privacy-utility tradeoff for applications using energy disaggregation of smart-meter data. In *Prov. Australasian Conference on Information Security and Privacy (ACISP)*, 2017.

[17] S. Iyengar, S. Lee, D. Irwin, P. Shenoy, and B. Weil. Watthome: A data-driven approach for energy efficiency analytics at city-scale. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[18] P. Jayaraman, X. Yang, A. Yavari, D. Georgakopoulos, and X. Yi. Privacy preserving internet of things: From privacy techniques to a blueprint architecture and efficient implementation. In *Future Generation Computer Systems*, May 2017.

[19] G. Kalogridis, C. Efthymiou, S. Denic, T. Lewis, and R. Cepeda. Privacy for Smart Meters: Towards Undetectable Appliance Load Signatures. In *SmartGridComm*, October 2010.

[20] W. Kleiminger, C. Beckel, T. Staake, and S. Santini. Occupancy detection from electricity consumption data. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, 2013.

[21] J. Z. Kolter and M. J. Johnson. Redd: A public data set for energy disaggregation research. In *Workshop on*

Data Mining Applications in Sustainability (SIGKDD), San Diego, CA, 2011.

[22] M. A. Lisovich, D. K. Mulligan, and S. B. Wicker. Inferring Personal Information from Demand-Response Systems. *IEEE Security and Privacy Magazine*, 8, November 2010.

[23] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse. The smart thermostat: using occupancy sensors to save energy in homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, 2010.

[24] P. McDaniel and S. McLaughlin. Security and Privacy Challenges in the Smart Grid. *IEEE Security and Privacy Magazine*, 33, November 2009.

[25] S. McLaughlin, P. McDaniel, and W. Aiello. Protecting consumer privacy from electric load monitoring. In *Proceedings of the 18th ACM conference on Computer and communications security*, 2011.

[26] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private Memoirs of a Smart Meter. In *Proceedings of ACM BuildSys*, November 2010.

[27] A. Panah, A. Yavari, R. van Schyndel, and D. Georgakopoulos. Context-driven granular disclosure control for internet of things applications. In *IEEE Transactions on Big Data.*, 2017.

[28] O. Parson, S. Ghosh, M. J. Weal, and A. Rogers. Non-intrusive load monitoring using prior models of general appliance types. In *AAAi*, 2012.

[29] S. N. Patel, T. Robertson, J. A. Kientz, M. S. Reynolds, and G. D. Abowd. At the flick of a switch: Detecting and classifying unique electrical events on the residential power line. In *International Conference on Ubiquitous Computing*, 2007.

[30] R. Perez-Pena and M. Rosenberg. Strava fitness app can reveal us military sites, analysts say. *The New York Times*, 2018.

[31] E. L. Quinn. Smart Metering and Privacy: Exisiting Law and Competing Policies. A Report for the Colorado Public Utilities Commission. Technical report, Sandia National Laboratories, April 2009.

[32] S. R. Rajagopalan, L. Sankar, S. Mohajer, and H. V. Poor. Smart meter privacy: A utility-privacy framework. *arXiv preprint arXiv:1108.2234*, 2011.

[33] Sense - energy monitor., 2019. https://sense.com/.

[34] E. M. Tapia, S. S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. In *International conference on pervasive computing*, 2004.

[35] X. Wang, D. Lei, J. Yong, L. Zeng, and S. West. An online load identification algorithm for non-intrusive load monitoring in homes. In *IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2013.

[36] W. Yang, N. Li, Y. Qi, W. Qardaji, S. McLaughlin, and P. McDaniel. Minimizing private data disclosures in the smart grid. In *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012.