Minimizing Private Data Disclosures in the Smart Grid

Weining Yang
Purdue University
yang469@cs.purdue.edu

Ninghui Li
Purdue University
ninghui@cs.purdue.edu

Yuan Qi
Purdue University
alanqi@cs.purdue.edu

Wahbeh Qardaji
Purdue University
wqardaji@cs.purdue.edu

Stephen McLaughlin
Penn State University
smclaugh@cse.psu.edu

Patrick McDaniel
Penn State University
mcdaniel@cse.psu.edu

ABSTRACT

Smart electric meters pose a substantial threat to the privacy of individuals in their own homes. Combined with non-intrusive load monitors, smart meter data can reveal precise home appliance usage information. An emerging solution to behavior leakage in smart meter measurement data is the use of battery-based load hiding. In this approach, a battery is used to store and supply power to home devices at strategic times to hide appliance loads from smart meters. A few such battery control algorithms have already been studied in the literature, but none have been evaluated from an adversarial point of view. In this paper, we first consider two well known battery privacy algorithms, Best Effort (BE) and Non-Intrusive Load Leveling (NILL), and demonstrate attacks that recover precise load change information, which can be used to recover appliance behavior information, under both algorithms. We then introduce a stepping approach to battery privacy algorithms that fundamentally differs from previous approaches by maximizing the error between the load demanded by a home and the external load seen by a smart meter. By design, precise load change recovery attacks are impossible. We also propose mutual-information based measurements to evaluate the privacy of different algorithms. We implement and evaluate four novel algorithms using the stepping approach, and show that under the mutual-information metrics they outperform BE and NILL.

Categories and Subject Descriptors

K.4.1 [**COMPUTERS AND SOCIETY**]: Privacy

General Terms

Security

Keywords

smart meter, privacy, load monitor

1. INTRODUCTION

The rapid replacement of traditional residential electric meters by networked smart meters has brought tangible concerns about electricity customer privacy [36]. Smart meters use solid state measurement circuits that can record minute- or second-level profiles of energy usage. In tandem with Non-Intrusive Load Monitoring (NILM), these fine-grained load profiles can be analyzed to reveal individual appliance usage [17], and ultimately reveal behaviors such as sleep patterns, number of occupants, and times of vacancy [28, 31]. Access to such sensitive information is not limited to utility providers. Multiple studies have shown smart meters to be vulnerable to attacks that could leak fine grained usage data to malicious third parties [13, 30], and load profiles may also be shared with third party data centers to give customers web access to their energy usage. Such concerns have caused public outcry, and let to smart meters being banned in multiple cities in North America and Europe [43].

One promising solution to privacy loss that does not require utility cooperation is that of Battery-based Load Hiding (BLH). BLH employs a battery to partially supply the net *demand load* from the house to alter the *external load* as seen by the smart meter. The battery is charged and discharged at strategic times to hide the load profile events caused by appliances being turned on and off. This removes the basic information needed by NILM algorithms to identify appliances, thus thwarting further analysis. The design of a battery control algorithm is a crucial part of a BLH system. BLH algorithms have to cope with limited battery capacity and discharge rates and hard to predict consumption patterns. The basic strategy taken by these algorithms is to flatten the load profile to a constant value as often as possible.

In this paper, we perform a new analysis of two well known BLH algorithms, Best Effort (BE) [20] and Non-Intrusive Load Leveling (NILL) [29], and show that they are vulnerable to previously unknown attacks that leak appliance events. These attacks are based on the way these algorithms handle *load peaks*, events of high demand loads that are beyond the battery's discharge rate. Such peaks can partially leak through into the external load, and sometimes can be fully recovered by analysis. Given insights from the failure of these algorithms to handle load peaks, we propose a different approach based on quantizing the demand load into a step function, where the step size is determined by the battery's maximum charge rate and maximum discharge rate. While previous quantization algorithms developed in the signal processing literature aimed to minimize the quantization error, our stepping algorithms aim to maximize it within battery limitations. This allows the stepping algorithms to better handle load peaks and to reveal less overall information as defined by mutual information metrics.

In doing this, we make the following contributions:

- We reveal new vulnerabilities in two existing BLH algorithms that allow for the recovery of substantial appliance

CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.

usage information. To the best of our knowledge, this is the first work to uncover flaws in BLH algorithms.

- We present a novel stepping-based framework for BLH algorithms based on maximizing error between the demand load and external load subject to battery capacity and charging/discharging rate constraints.

- We evaluate the four stepping algorithms against the NILL and BE algorithms on real-world energy consumption, and show that all of them generally outperforms NILL and BE, and lazy stepping algorithms consistently outperform other algorithms.

The insight behind the stepping approach is as follows. We observe that the measured time series data have two dimensions: time and value, and the privacy threat caused by smart meters is due to finer-grained measurement in the time dimension. The key idea of the stepping framework is to make the value dimension more coarse-grained. The stepping approach is similar to the idea of quantization in signal processing, which is the process of mapping a large set of input values to a smaller set — such as rounding values to some unit of precision. Because quantization is a many-to-few mapping, it is an inherently non-linear and irreversible process.

In the stepping approach, the algorithm forces the external load to be multiples of β, a value chosen based on the battery's parameters. This results in the external load being a step function. Given a demand load value, a stepping algorithm decides whether to force the external load to the level above the demand by charging the battery or to the level below the demand by discharging it.

We consider three different kinds of stepping algorithms. Lazy_Stepping algorithms try to maintain the external load unchanged as long as possible. The Lazy_Charging algorithm tries to keep charging the battery until is full and then keep discharging the battery until it is empty. Random_Charging algorithms randomly choose whether to choose to charge or discharge the battery. Because the stepping approach effectively maximizes the error between demand load and external load, when one observes a load-change event in the external load, estimating the amplitude of the change has an uncertainty range of 2β.

We have conducted extensive experiments, using two data sources. One source is the data used in [29], which consists of one-second measurement of data in four houses over a few months. The other is a dataset [39] that includes electricity data measured at one-minute resolution in 22 dwellings over two complete years (2008 and 2009). To measure the amount of information leakage and compare different stepping algorithms as well as comparing with BE and NILL, we use several mutual information measures.

We note that the problem is a special case of changing a time-series data to protect privacy. This setting is different from privacy-preserving data publishing, where one hides the existence of one item. Here, the whole time series data belongs to one individual and needs privacy protection. Similar problems also occur in other domains such as medical sensors, which can produce time series data that have privacy implications [18, 1].

The remainder of this paper is organized as follows. Section 2 provides background on smart meters and NILM algorithms. Section 3 defines the problem we are solving. Section 4 shows attacks against BE and NILL that leak appliance information. Section 5 details the stepping approach and algorithms. Section 6 presents evaluation using mutual information metrics. Finally, Section 7 covers related work in smart meter privacy, and Section 8 concludes. An Appendix gives formulas for computing mutual information measures.

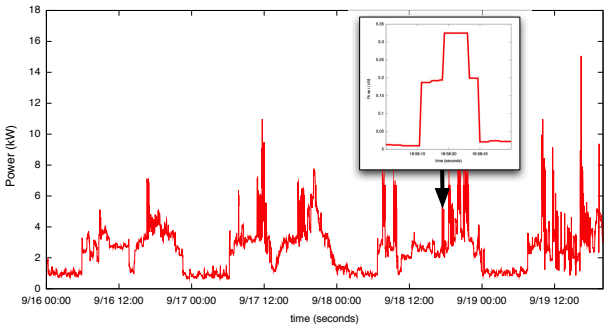


Figure 1: An example load profile with appliance events.

2. BACKGROUND

Smart Electric Meters. Smart electric meters are networked embedded systems that are currently replacing traditional electromechanical residential meters throughout the world. Smart meters promise novel features such as dynamic pricing schemes [23], remote meter reading [10], improved power outage reporting [21], and load curtailment in emergencies [14]. As a result of their enhanced measurement and storage capabilities, smart meters can maintain fine-grained time series of electricity measurements. These are known as *load profiles*. Unlike traditional power meter measurements, which are at the granularity of months, load profiles can be at the granularity of minutes or seconds, making it possible to discern individual rising and falling edges in net energy consumption.

An example of this is shown in Figure 1, which is adapted from [29]. The figure depicts a four day load profile of a single residence. A specific set of features are highlighted in the smaller box. These features show an entertainment system, and subsequently a television turning on, then turning off in reverse order. Given this type of data, individual appliance events can be extracted as described in the following section.

Nonintrusive Load Monitoring. Nonintrusive (Appliance) Load Monitory (NILM) is a technique for analyzing a household's net electric load profile in order to deduce what electric appliances are being used [16]. The time resolution of these profiles may be on the order of minutes or seconds, all of which can be obtained using commercially available smart meters. NILM is *nonintrusive* in the sense that individual appliances need not be instrumented. Hence, such appliance monitoring can be done remotely and without the knowledge of the household residents.

While there have been numerous approaches to NILM [40, 12, 33, 6, 7, 15, 11, 34, 8, 26], the most successful in residential settings are based on *edge-detection* [16, 25]. Edge-detection techniques look for significant changes in the steady state current being consumed by the household. Such changes are characterized by sharp *edges* in the electric current consumed by the household. These edges are then clustered and matched against known appliance profiles. For instance, if someone in the household turns on a 40 Watt lamp, then the net current increases by 40 Watts. Conversely, when the lamp is turned off, the net current drops by the same amount. The NILM algorithm will detect the pair of edges with equal magnitude and opposite direction, and match them against the electric profile for a 40 Watt lamp.

While NILM algorithms do have useful applications such as informing electricity customers about their usage patterns and allowing utilities to verify that customers honor load reduction agree-

Battery parameters	*Meaning*
$\mathbf{C_H}$	The upper safe limit on the battery's state of charge
$\mathbf{C_L}$	The lower safe limit on the battery's state of charge
β_c	The maximal rate at which the battery can be charged.
β_d	The maximal rate at which the battery can be discharged.
Time series data	
$t = 0, 1, 2, \cdots$	This denotes time.
$d(t)$	The aggregated net demand, which we call the demand load. This is the input into a BLH algorithm.
$d'(t) = d(t) - d(t-1)$ for $t \geq 1$	The difference time series of demand load.
$e(t)$	The load after mixing in the battery's power, which we call the external load; this is the load drawn from the power grid. This time series is determined by the algorithm.
$e'(t) = e(t) - e(t-1)$ for $t \geq 1$	The derivative of the external load $e(t)$.
$b(t)$	The battery's rate of charge.
$C(t)$	The battery's state of charge.
Constraints due to battery	
$e(t) \geq 0 \wedge b(t) = e(t) - d(t)$	Here we assume an idealized battery model, and require that there is no wasted energy.
Rate limit: $\beta_c \geq b(t) \geq -\beta_d$	The charging and discharging rate must be within the battery's parameters.
$C(t) = \sum_{i=1}^{t} b(i) + C(0)$	$C(0)$ is the battery's initial state of charge
Capacity limit: $\mathbf{C_L} \leq C(t) \leq \mathbf{C_H}$	The battery's state of charge should be within the battery's upper and lower safe limit.

Table 1: Notations for battery parameters, time series data, and constraints.

ments [9], they also pose a clear threat to household privacy [24, 36]. In one study, it was shown that typical appliances for cooking breakfast could be distinguished from the net load [31], and another was able to accurately detect household presence and sleep cycles [28]. Potential misuses of this data include targeted advertising [35], predatory pricing [4], and potential leakage to malicious third parties such as organized crime. NILM algorithms have also been tuned to detect usage of dryers [27], space heaters [15], and energy efficient appliances [33]. It was also suggested early on in the development of NILMs that they may be used for surveillance purposes [17]. Of course, proliferation of smart meters opens the public to both the benefits and drawbacks of NILM.

3. PROBLEM DEFINITION

In Battery-based Load Hiding (BLH), one connects a rechargeable battery between the smartmeter and the internal wire. Power from the battery is mixed in to provide for the electricity demand of the house, adjusting the amount of electricity the house draws from the electric grid. Thus one can partially disguise the actual load to reduce the amount of information leakage from smart meter readings.

We now describe the time series data in the model, which are also given in Table 1. Two key time series data are the **demand load** $d(t)$, which represents the electricity demand at time t, and the **external load** $e(t)$, the load drawn from the power grid and observed by the smart meter. While conceptually the time ranges over nonnegative real numbers, measurement can be made only at fixed time intervals; thus we have t range over non-negative integers.

In this paper, we assume that $e(t) \geq 0$; that is, the home energy system cannot sends electricity to the grid. This assumption is consistent with our model of using a battery. If one has a home generator and a more sophisticated home energy system, then this assumption can be removed. We do not expect removing this assumption to significantly affect the analysis of existing algorithms and the privacy protection effectiveness result of the algorithms proposed in this paper. We also abstract away details about the battery and assume an idealized battery model, i.e., charging and discharging the battery has no energy waste.

Based on these assumptions, the difference between the two time series $e(t)$ and $d(t)$ is provided by the battery. We write this as $b(t) = e(t) - d(t)$. When $b(t) > 0$, the battery is charging at the rate $b(t)$; and when $b(t) < 0$, the battery is discharging to provide electricity at the rate of $|b(t)|$ to meet the demand.

The times series $b(t)$ must satisfy the **rate limit** constraint that $b(t)$ must be within the range allowed by the battery, i.e., $\beta_c \geq b(t) \geq -\beta_d$, where β_c is the maximal charging rate, and β_d is the maximal discharging rate. The battery's state of charge $C(t)$, is given by $C(t) = \sum_{i=1}^{t} b(i) + C(0)$, where $C(0)$ is the battery's initial state of charge. The battery state must satisfy the **capacity limit** constraint $\mathbf{C_L} \leq C(t) \leq \mathbf{C_H}$, where $\mathbf{C_L}$ and $\mathbf{C_H}$ give the lower and upper safe limits of the battery's state of charge.

A Battery-based Load Hiding (BLH) algorithm needs to compute the external load $e(t)$ based on the demand $d(t)$ and the battery's state such that the constraints are satisfied, while minimizing the amount of information that can be inferred from $e(t)$.

Load changes reflect potentially sensitive private information, because it reflects user behavior. Furthermore, most NILM approaches also compute load changes as the first step for performing analysis. Therefore, we define the demand load change time series as $d'(t) = d(t) - d(t-1)$, and the external load change time series as $e'(t) = e(t) - e(t-1)$. Note that one could use the load change time series to recover the original time series, with knowledge of the overall average.

The primary goal of BLH algorithms is to prevent leaking of information in the demand load change series $d'(t)$ from observing $e(t)$, or equivalently observing $e'(t)$. To prevent inferencing of $d'(t)$ from observing $e(t)$, one approach is to avoid changes in $e(t)$ as much as possible. Indeed, if $e(t)$ is constant, then no information beyond the average energy usage is leaked. This is possible only with a very large battery. The key design decision for BLH algorithms is how to handle the case when the battery's limitation means that $e(t)$ can no longer be held constant.

To show that a particular algorithm leaks information, it suffices to find one method to recover demand load change information from the output produced by the algorithm.

To evaluate algorithms when no obvious attack exist, we must develop metrics that are independent of particular attacks. We ob-

serve that $e(t)$ must be somewhat correlated with $d(t)$ because of the rate limit and capacity constraints. We use the mutual information between the two time series $\mathbf{e}' = [e'(1), \ldots, e'(T)]$ and $\mathbf{d}' = [d'(1), \ldots, d'(T)]$ where T indexes the last time point in $\mathbf{e}'$ and $\mathbf{d}'$.

Intuitively, mutual information between two random variables X and Y measures the information that X and Y share: it measures to what extent knowing one of these variables reduces uncertainty about the other. For example, if X and Y are independent, then knowing X does not give any information about Y and vice versa, so their mutual information is zero. At the other extreme, if X and Y are identical then all information conveyed by X is shared with Y: knowing X determines the value of Y and vice versa. As a result, when X and Y are identical the mutual information is the same as the uncertainty contained in Y (or X) alone, namely the entropy of Y (or X—clearly if X and Y are identical they have equal entropy).

Specifically, we consider three mutual information measures between $\mathbf{e}'$ and $\mathbf{d}'$. The first one models the joint states $\{f(t) = [e'(t)d'(t)]\}$ as independent points for each t, and measures the mutual information. This measures to what extend a load change in $e'(t)$ is correlated with a load change in $d'(t)$. The second one converts $e'(t)$ and $d'(t)$ into a binary series; this measures to what extend the information of whether a load changes in $e'(t)$ is correlated with whether a load change occurs in $e'(t)$. In the third one, we mode joint states $\{f(t) = [e'(t)d'(t)]\}$ as a first-order Markov chain and measure mutual information between $e'(t)$ and $d'(t)$. Because of the time series nature of the data, the Markov chain assumption is more realistic than the independence assumption, and possibly can better measure the information leak from $e'(t)$ to $d'(t)$. In the Appendix, we present formulas for computing the mutual information measures.

4. ANALYZING EXISTING ALGORITHMS

In this section, we present highly accurate methods to recover load-change events from output produced by the Best Effort algorithm in [20], and the NILL algorithm in [29].

4.1 The Best Effort Algorithm

Kalogridis et al. [20] proposed a best-effort algorithm, which we call BE in this paper. This algorithm tries to avoid changing the external load $e(t)$ whenever possible, and when the actual demand $d(t)$ differs from $e(t)$, we charge or discharge the battery to make up the difference. There are four cases when the battery cannot make up the difference, and $e(t)$ has to change. After each change, the algorithm tries to maintain the new external load until the next time one of the four cases occurs. These four cases are given in Table 2.

Cases 1 and 2 are due to battery capacity constraints. The first case is when maintaining the load will overcharge the battery. When one keeps $e(t) = e(t-1)$, then the battery's state at time t will be $C(t-1) + e(t-1) - d(t)$. If this is greater than $\mathbf{C_H}$, then one cannot maintain $e(t) = e(t-1)$. Observe that because $C(t-1) \leq \mathbf{C_H}$, when $C(t-1) + e(t-1) - d(t) > \mathbf{C_H}$, it must be $e(t-1) > d(t)$. In this case $e(t)$ is set to $d(t)$; and thus $e(t) > d(t)$, and we have an observable load increase. The second case is when maintaining the load will cause the battery's state of charge to be too low, i.e., when $C(t-1)+e(t-1)-d(t) < \mathbf{C_L}$. In this case, the BE algorithm also sets $e(t)$ to be the current demand $d(t)$, and this results in an observable load decrease.

Cases 3 and 4 are due to battery rate constraints. They are checked only when Cases 1 and 2 are not triggered. When either Case 1 or 2 is triggered, we have $e(t)$ set to $d(t)$, and the battery rate constraints are trivially satisfied. The third case is when maintaining the external load results in wasted energy even when maximally charging the batter, i.e., $e(t-1) > d(t) + B_c$. In this case, BE drops the external load to the level of providing for the demand while maximally charging the battery, i.e., $e(t)$ is set to be $d(t) + \beta_c$. The fourth case is when maintaining the external load cannot provide for the demand even when maximally discharging the batter, i.e., $e(t-1) < d(t) - B_d$. In this case, BE increases the external load to provide for the demand with maximally discharging from the battery, that is, setting $e(t)$ to be $d(t) - \beta_d$.

4.2 Information Leakage in the BE Algorithm

We observe that each time the load changes, the new load $e(t)$ depends on the the demand $d(t)$. When a load increase occurs ($e(t) > e(t-1)$), we know that either $e(t) = d(t)$ (case 2), or $e(t) = d(t) - \beta_d$ (case 4).

Recall that we are concerned primarily with inference of load-change information. Knowing $d(t)$ at a single time point does not leak load-change information. We need two consecutive load changes to attempt to recover a load-change event. As it turns out, in the external load produced by the BE algorithm, about half of load changes occur after another change. Furthermore, almost all consecutive changes have the same up or down direction. That is, they are of the form (up,up) or (down,down), as opposed to (up,down) or (down,up). Finally, when the two consecutive changes are in the same direction, they are almost always from the same case. That is, two (up,up) changes are either both due to case 2, or both due to case 4. As a result, this yields a highly-accurate method of predicting load-changes from the output produced by the BE algorithm. The algorithm looks for two consecutive changes of the same direction, i.e., $e(t-1) < e(t) < e(t+1)$ or $e(t-1) > e(t) > e(t+1)$, and then predicts that at time $t+1$ there is a load change in the demand load of the magnitude of $e(t+1) - e(t)$.

These features are due to the following reasons. When an energy event starts, there are often consecutive load increases in $d(t)$, and when the event ends, there are often consecutive load decreases in $d(t)$. When an increase in $d(t)$ triggers case 2, an increase occurs in $e(t)$ and $e(t) = d(t)$. In this case, the battery is low, and a further increase in $d(t+1)$ would thus also trigger an increase at time $t+1$ with $e(t+1) = d(t+1)$. Similarly, when an increase in $d(t)$ triggers case 4, the demand $d(t)$ is too high, and the BE algorithm sets $e(t)$ so that the battery is maximally discharging. When $d(t+1) > d(t)$, then we again have case 4 occurring at time $t+1$, causing two consecutive increases of the same case.

We evaluate the effectiveness of this edge inference in Section 6.

4.3 The NILL Algorithm

The NILL algorithm in [29] has three states, and attempts to maintain a different constant load for each state. The algorithm is illustrated in Table 3, and described below.

The Stable (ST) State. The algorithm starts in this state. In this state, the algorithm attempts to set the external load to K_{ST} when possible, where K_{ST} is the algorithm's guess of the average load in the near future, and is adjusted with state changes.

The algorithm needs to leave the ST state when one of the following two situations occurs. The first is when maintaining K_{ST} will overcharge the battery, in which case it goes to the High Recovery (HR) State. The second is when maintaining K_{ST} will cause the

Case	Condition for Changing e(t)	BE Behavior	Observable Change
1 (batter full)	$C(t-1)+e(t-1)-d(t) > \mathbf{C_H}$	$e(t) \leftarrow d(t)$	load decrease (down)
2 (battery low)	$C(t-1)+e(t-1)-d(t) < \mathbf{C_L}$	$e(t) \leftarrow d(t)$	load increase (up)
3 (demand drops)	$e(t-1) > d(t)+\beta_c$	$e(t) \leftarrow d(t)+\beta_c$	load decrease (down)
4 (demand increases)	$e(t-1) < d(t)-\beta_d$	$e(t) \leftarrow d(t)-\beta_d$	load increase (up)

Table 2: The cases when the Best Effort algorithm must change its output. Cases 1 and 2 are due to battery capacity constraints. Cases 3 and 4 are due to battery rate constraints.

State	Case	Condition	NILL Behavior	Observable Load Change
ST	S1 (battery full)	$C(t-1)+K_{ST}-d(t) > \mathbf{C_H}$	$s \leftarrow$ HR; $K_H \leftarrow d(t)-0.5$AMP	decrease to K_H
ST	S2 (battery low)	$C(t-1)+K_{ST}-d(t) < \mathbf{C_L}$	$s \leftarrow$ LR; $K_L \leftarrow \beta_c$	increase to K_L
ST	S3 (demand high)	$d(t) > K_{ST}+\beta_d$	$e(t) \leftarrow d(t)-\beta_d$	$e(t) > K_{ST}$, leaks $d(t)$
ST	S4 (demand low)	$d(t) < K_{ST}-\beta_c$	$e(t) \leftarrow d(t)+\beta_c$	$e(t) < K_{ST}$, leaks $d(t)$
ST	S5 (normal ST)	True	$e(t) \leftarrow K_{ST}$	flat at K_{ST}
HR	H1 (battery high)	$C(t-1)+K_H-d(t) > \mathbf{C_H}$	$K_H \leftarrow d(t)-0.5$AMP; $e(t) \leftarrow K_H$	decrease to new K_H
HR	H2 (discharged enough)	$C(t-1) < 0.5\mathbf{C_H}+0.5\mathbf{C_L}$	$s \leftarrow$ ST; $K_{ST} \leftarrow 0.5K_{ST}+0.5Avg$	goes to ST
HR	H3 (demand high)	$d(t) > K_H+5$AMP	$s \leftarrow$ ST; $K_{ST} \leftarrow 0.5K_{ST}+0.5Avg$	goes to ST
HR	H4 (normal HR)	True	$e(t) = K_H$	flat
LR	L1 (charged enough)	$C(t-1) > 0.8\mathbf{C_H}+0.2\mathbf{C_L}$	$s \leftarrow ST$; $K_{ST} \leftarrow 0.5K_{ST}+0.5Avg$	goes to ST
LR	L2 (demand high)	$d(t) > \beta_c$	$e(t) \leftarrow d(t)$	$e(t) > K_L$, leaks $d(t)$
LR	L3 (normal LR)	True	$e(t) \leftarrow \beta_c$	flat at $K_L = \beta_c$

Table 3: The NILL algorithm. At each time t, the algorithm follows the cases for the current state sequentially; if a state change occurs, the algorithm follows the cases in the new state to determine $e(t)$. Observe that in S3, S4, and L2, information about $d(t)$ is leaked.

battery to be too low, in which case it goes to the Low Recovery (LR) State.

Even when the algorithm is in the ST state, it may be infeasible to maintain the external load to be K_{ST}. The demand may be either too high (Case S3), or too low (Case S4). In these cases, the external load is set to be using the battery to the maximal possibility.

When the system returns to the ST state from either HR or LR, the value K_{ST} is updated to be a weighted average of the most recent K_{ST} and Avg, the average load during the most recent state. That is, the new K_{ST} is set to be $\alpha Avg + (1-\alpha)K_{ST}$, where α is chosen to be 0.5 in the experiments in [29].

The High Recovery (HR) State. In this state, the system should be drawing an external load lower than the demand, and gradually discharging the battery. When entering the HR state, the NILL algorithm sets $e(t)$ to be K_H, which is chosen to be 0.5Amp lower than the most recent demand, and when this is lower than the new demand, then it resets K_H. The NILL algorithm returns to the ST state when the demand load is 5Amp higher than the current K_H or when the battery is discharged to 50% of the capacity.

The Low Recovery (LR) State. In this state, the system should be drawing an external load that is higher than the demand $d(t)$, and gradually charging the battery. The NILL algorithm sets $e(t)$ to be the max charging rate B_c, and returns to the ST state when the battery is charged to 80% of the capacity.

4.4 Information leakage in the NILL Algorithm

As can be seen from Table 3, in three cases the NILL algorithm's output $e(t)$ depends on $d(t)$: Cases S3, S4, and L2. Case S4 almost never occurs, as unless the battery's charging rate is really small compared with the average load, we have $K_{ST}-\beta_c < 0$, and Case S4 cannot happen. Both S3 and L2 are due to the demand being too high, which we call load peaks or just peaks. In the stable state, a demand $d(t)$ is a peak if $d(t) > \beta_d + K_{ST}$, and in this case $e(t)$ is set to $d(t)-\beta_d$. In the lower recovery state, a demand $d(t)$ is a peak if it is higher than β_c, and $e(t)$ is set to $d(t)$. Thus if we can identify when these peaks occur and which state the system is in, we can accurately recover the demand at these times. When we can recover two consecutive $e(t)$'s, we can recover a load-change event.

Identifying which state the system is in from the output is feasible. The K_{ST} values for each stable state period is easily identified, since it is maintained for an extended period of time. Furthermore, when the external load goes from K_{ST} up to the maximal charge rate, this indicates that the battery enters the LR state. When the external load goes down from K_{ST}, this indicates that the battery enters the HR state.

Discovering peaks is also feasible. As the NILL algorithms tries to maintain different stable loads in each state, every load higher than the stable load for the current state is a peak. If we find two peak $e(t)$ readings consecutive, then we can recover a load-change event with accurate magnitude.

One issue that complicates the recovering is that a state change from ST to LR may occur during a peak period. As a peak demand load during the stable state draws maximum discharge rate from the battery, a sustained peak period will cause the system to enter the LR state. Before the state change, we have $e(t) = d(t)-\beta_d$, and after the state change we have $e(t) = d(t)$. From observing that the external load values before and after the peak period, we can discover that such a state change has occurred during the peak period; however, we do not know exactly when the state change occurs. We observe that when we infer load-change events, not accounting for the state change will result in only one inaccurate load-change event prediction during the peak period, because we are subtracting two consecutive predicted demand loads. Our algorithm tries to predict the state change time when the confidence is high. When among all load changes during the peak period, only

New Time series data	
$h(t) : e(t) = h(t)\beta; \beta = \min(\beta_c, \beta_d)$	The series $h(t)$ determines the external load $e(t)$.
$s(t) = \begin{cases} 1 & \text{if } e(t) \geq d(t) \\ 0 & \text{otherwise} \end{cases}$	The charging signal.

Alg	Summary of algorithm
LS1	Keep $e(t) = e(t-1)$ if possible; otherwise, randomly chooses $s(t) \leftarrow \{0, 1\}$.
LS2	Keep $e(t) = e(t-1)$ if possible; otherwise, chooses $s(t) \leftarrow 1$ if and only if $C(t) < (\mathbf{C_H} + \mathbf{C_L})/2$.
LC	Keep $s(t) = s(t-1)$ if possible; otherwise, $s(t) \leftarrow 1 - s(t-1)$.
RC	Randomly chooses $s(t)$ each time, where $\Pr[s(t) = 0] = \frac{C(t) - \mathbf{C_L}}{\mathbf{C_H} - \mathbf{C_L}}$.

Table 4: Time series for the Stepping Framework

one such change is very close to an increase of β_c, then the algorithm predicts that the state change occurs at this time. If no such time is found, the algorithm does not predict the state change.

We evaluate the effectiveness of this load change recovering algorithm in Section 6.

5. THE STEPPING FRAMEWORK

We propose a novel algorithmic framework for BLH algorithms, which we call the stepping framework. We observe that the measured time series data have two dimensions: time and value, and smart meters result in finer-grained measurement in the time dimension, casing privacy threats. The stepping framework compensates for that by making the value dimension more coarse-grained via quantization. Because quantization is a many-to-few mapping, it is an inherently non-linear and irreversible process. As the same output value is shared by multiple input values, it is impossible in general to recover the exact input value when given only the output value.

The stepping framework is illustrated in Table 4. In the stepping framework, the algorithm makes the external load to be multiples of β, a value chosen based on the battery's parameters. Thus, $e(t)$ can be defined as $e(t) = h(t)\beta$, where $h(t)$ always takes integer values, and the shape of $e(t)$ will look like a step function; hence the name.

We choose the step value β to be the largest value that is feasible, i.e., for any possible demand load $d(t)$, there exists an integer h such that maintaining $h\beta$ satisfies the battery's capacity and rate constraints. Such a value is given by $\beta = \min(\beta_c, \beta_d)$. This is feasible, as for any $d(t)$, one can choose either the level just higher than $d(t)$ and charge the battery at a rate $\leq \beta_c$, or choose the level just lower than $d(t)$ and discharge the battery at rate $\leq \beta_d$. When the battery's state is close or at $\mathbf{C_H}$, then one chooses the lower level; and when the battery's state is close to or at $\mathbf{C_L}$, one chooses the higher level. Any larger β value is no longer feasible. For example, if $\beta > \beta_c$, then when the battery is at $\mathbf{C_L}$ and $d(t) = \beta + (\beta - \beta_c)/2$, setting $e(t)$ to be β or lower is not feasible because it requires to keep discharging the battery, and setting $e(t)$ at 2β or higher is not feasible because this requires charging the battery at a rate at least $(\beta + \beta_c)/2 > \beta_c$. Similarly, when $\beta > \beta_d$, then when the battery is at $\mathbf{C_H}$, and $d(t) = \beta - (\beta - \beta_d)/2$ cannot be feasibly provided.

5.1 Different Stepping Algorithms

To satisfy the rate limit constraint, it suffices to ensure that the algorithm always chooses among the two adjacent levels that sandwich $d(t)$ between them. Therefore, given a demand load, a stepping algorithm only needs to decide whether to choose the level that is higher than the demand load (in which case the battery is charging) or the level that is lower than the demand load (in which case the battery is discharging). We use $s(t)$ to denote whether the battery is charging or not. That is $s(t) = 1$ when $d(t) < h(i) * \beta$, and $s(t) = 0$ when $d(t) \geq h(t) * \beta$. We call $s(t)$ the charging signal. We note that when $d(t)$ is given, the charging signal $s(t)$ uniquely determines the external load $e(t)$. When $s(t) = 1$, we have $h(t) = \left\lceil \frac{d(t)}{\beta} \right\rceil$. When $s(t) = 0$, we have $h(t) = \left\lfloor \frac{d(t)}{\beta} \right\rfloor$ when $d(t)$ is not a multiple of β, and $h(t) = \frac{d(t)}{\beta} - 1$ when $d(t)$ is a multiple of β. Therefore, an algorithm in the stepping framework can be specified by describing how the charging signal $s(t)$ is determined.

Many algorithms are possible in the stepping framework. We consider the following stepping algorithms.

Lazy_Stepping (LS1 and LS2). Lazy stepping algorithms try to maintain $e(t)$ unless it is pushed to change. There are three cases in which changes must occur: (1) When maintaining the load results in overcharging the battery, in which case $s(i)$ must be set to 0; (2) When maintaining the load results in low battery, in which case $s(i)$ must be set to 1; and (3) when $d(t)$ is either too low or too high for $e(t-1)$, i.e., $d(t) \leq (h(t-1)-1)*\beta$ or $d(t) \geq (h(t-1)+1)*\beta$. We consider two alternatives, which we call LS1 and LS2. LS1 sets $s(i)$ to be 1 if the battery is below half, and 0 otherwise. LS2 sets $s(i)$ randomly to be 1 or 0.

Lazy_Charging (LC). In this algorithm, one tries to maintain the charging signal $s(t)$ unchanged unless the battery is either too low or too high, in which case one changes $s(t)$. One advantage of the LC algorithm is that this reduces the number of charge/discharge cycles for the battery, unlike all other algorithms we have seen so far. One disadvantage of the LC algorithm is that it is generally easy to predict $s(t)$, reducing the possible range of $d(t)$ from 2β to β.

Random_Charging (RC). In this algorithm, one sets $s(t)$ independent of $d(t-1)$ and $s(t-1)$. In RC, we set the probability $s(t)$ is set to 1 is determined by the battery's state: $\Pr[s(t) = 0] = \frac{C(t) - \mathbf{C_L}}{\mathbf{C_H} - \mathbf{C_L}}$.

5.2 Information Leakage of Stepping Algorithms

One strength of the stepping framework is that no matter which specific stepping algorithm one uses, one always obtains some degree of privacy guarantee.

We observe that it is sometimes possible to predict $s(t)$. A trivial example is that observing $e(t) = 0$ one knows for sure that $s(t) = 0$. Accurately predicting $s(t)$ is also possible in LC. However, as argued below, even if one could accurately predict $s(t)$, after observing $e(t)$, the possible range of $d(t)$ is still of size β.

We note that the stepping algorithms differ from existing quantization algorithms used in signal processing because existing work

aims at minimizing quantization error, and our privacy protection goal means that we want to maximize such error, subject to the restriction that one can quantize a value only to the upper or lower integer values. One difference is as follows. When the demand load is 2.9β, reducing quantization error implies choosing 3β. However, none of the stepping algorithms proposed above would prefer 3β to 2β. As a result, when one observes that the load is 2β at time t, one knows that the demand load is in $(\beta, 3\beta)$. Even if one can predict that the battery is currently discharging, the possible range of the actual demand load at time t is still $[2\beta, 3\beta)$. Furthermore, when one observes that next time instant $(t+1)$ the external load is 4β, one knows that a load increase event has occurred; however, even if one could tell that the battery is still discharging, the possible range of demand load at $t+1$ is still $[4\beta, 5\beta)$, thus the possible range for the increase amplitude is $(1\beta, 3\beta)$, and the observer's uncertainty range is 2β.

Therefore, highly accurate load change detection, such as what we can do for BE and NILL, is impossible for stepping algorithms. However, it is still possible to recover some information about demand load changes from the external load changes. We use mutual information measures to evaluate these stepping algorithms.

6. EXPERIMENTAL RESULTS

6.1 Datasets and Experiment Methodology

In our experiments, we use two collections of datasets: one-second resolution datasets and a one-minute resolution datasets. The one-second resolution datasets were collected in four houses and apartments in the north-eastern United States over the course of one month in spring. Specific details of these datasets can be found in [29]. The one-minute resolution datasets are from a study published on the UK Data Archive [39]. This study includes electricity data measured at one-minute resolution in 22 dwellings over two complete years (2008 and 2009). Each dwelling was fitted with a single meter covering electricity use of the whole dwelling.

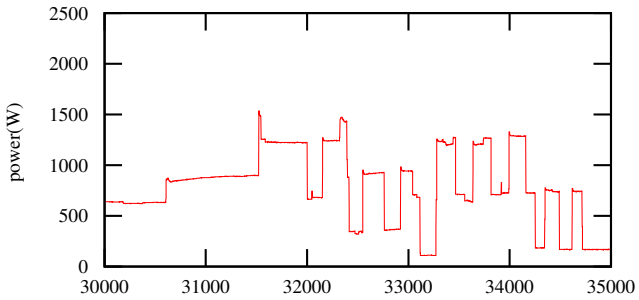
We note that some of the data are not complete. Small gaps in the one-second dataset were patched using interpolation as described in [29]. We refer to these four datasets as S1 through S4.

There also exist some holes where data is missing in the one-minute data. For each of these datasets, we extract the longest consecutive sets of measurements. We chose the six longest segments to use, and denote them by M1 through M6. The longest, denoted by M1, is a full year long. The shortest is about 244 days long.

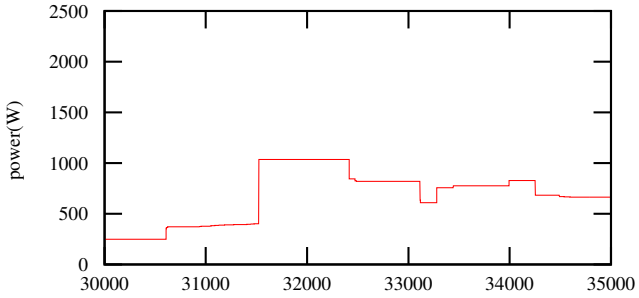
In each experiment, we evaluate batteries of different sizes. For ease of interpretation, we report battery sizes in Kilowatt hours (KWh)[1] For example, a 1.0 KWh battery can be thought of as delivery 1 KW of energy for 1 hour before being depleted. In our experiments, we assume $\beta_c = \beta_d$ and all the batteries have the same ratio between capacity and β_d. Specifically, we set max discharge rate at 1C, which means that if a full battery discharges at its maximum rate, it will be discharged to empty in 1 hour.

Before presenting the numerical experimental results, we first plot the external load outputted by BE, NILL and LS2 on a segment of dataset S1, in order to give an intuitive feeling of these algorithms' behavior. Figure 2 shows the plot; it includes the original demand load at the top to felicitate examination. The battery used has capacity 0.5KWh. We can clearly see that the behavior of NILL is such that all demands lower than 0.5KW is hidden, but demand loads above 0.5KW are mostly preserved in the output. In the output of BE, we observe that the slope from around 30600 to

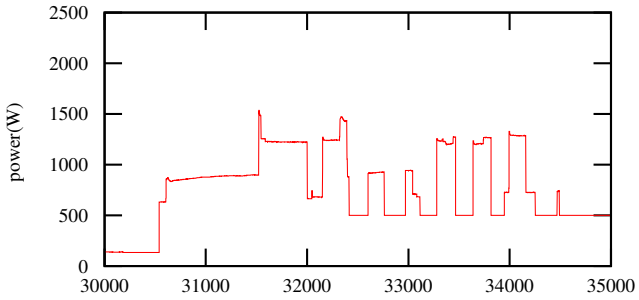
[1]While batteries are often specified in terms of Amp hours (Ah), the results are equivalent.



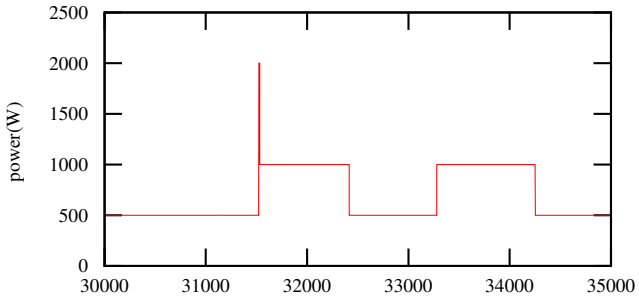
(a) Original demand load



(b) External load by BE



(c) External load by NILL



(d) External load by LS2

Figure 2: Effect of different algorithms on dataset S1 with battery capacity 0.5KWh.

31600 clearly match that in the original demand. However, the LS2 stepping algorithm leaks very little information from the demand series.

6.2 Load-Change Detection of NILL and BE

We now evaluate the efficacy of our analysis described in Section 4 to detect events in external loads under both NILL and BE. As the stepping algorithms are by design not vulnerable to this analysis, they are evaluated later using mutual information measures.

Table 5 reports the results of detecting load-change events in the output of NILL and BE algorithm. As can be seen, significantly more events can be recovered under NILL than under BE. This is because when NILL encounters a period of peak loads it leaks the shape of the demand load during this period. BE, on the other hand, attempts to maintain the current load at all times, and thus only part of the beginning and the ending of the peak periods can be detected. However, for both algorithms, the detection technique was satisfying. For one-second datasets, we got 96%-100% accuracy on identifying detected load events for almost all our experiment.

Focusing on the left hand side of the table, it is clear that both algorithms are dependent upon battery size. This is because larger batteries have higher discharge rates ($\beta_c/\beta_d = 1$), and thus more peaks can be completely covered by both algorithms. Turning to the right hand side, it can be seen that the accuracy of our detection algorithm is independent of the nature of the residence, e.g., many events or few events. In light of this, we conclude that both of these error-minimizing algorithms are insufficient to hide appliance usage without a prohibitively expensive battery.

We observe that the detection results for the one-minute dataset corroborate with those for the one-second data. While the numbers of detected events are smaller, this is due to the fact that one day has 1/60 as many minutes as the number of seconds. When adjusted by a factor of 60, the percent of time points which are detected is significantly higher than that in the one-second. This higher detection rate, however, comes at a cost of slightly lower accuracy rate, which ranges from about 80% to about 95%. These results show that even at the lower resolutions used by modern smart meters, the accuracy of the detection algorithm is still acceptable, albeit, fewer individual load events are detectable because of the lower resolution.

6.3 Evaluating Stepping Algorithms

We now turn to our comparison of the stepping algorithms with the NILL and BE algorithms under the mutual information metrics and . The results for minute-level datasets are shown in Figure 3.

Figures 3(a) and 3(b) show mutual information under the independence assumption. As can be seen in 3(a), the mutual information is highly dependent on battery size. This is due to the same effect seen in the previous section, where fewer partial events leak into the external load with a higher maximum discharge rate. Figure 3(b) shows that for a 0.5 KWh battery[2], three of the stepping algorithms LS1, LS2 and RC clearly dominates the other algorithms on all four residences.

While stepping algorithms can naturally hide magnitude of load changes, we are also interested in knowing whether they can hide the existence of change events. To do this we repeat the experiment, this time we set a threshold, which is 20W in the experiment. Any value in $e'(t)$ or $d'(t)$ above the threshold are considered to be an event and thus assigned as 1 and any value below as 0. As can be seen in Figures 3(c) and 3(d), for the given threshold value, these results are significantly less dependent on battery size. Also in contrast to the previous experiment, the RC algorithm outperforms the others. This is because whether there is a change or not in RC's output is randomly determined; thus it is less correlated with $d(t)$. However, LS1 and LS2 still outperforms all algorithms except RC.

The results for the mutual information under the Markov assumption are shown in Figure 3(e) and 3(f). The results are close to those with the independence assumption, with LS1 and LS2 clearly dominating other algorithms. This suggests that the dependencies between load events are significantly weakened by load hiding.

The results for mutual information of the one-second data are shown in Figure 4. We can see that the results are similar to that of the one-minute data, except that for one of four datasets (S4), BE performs the best for mutual information and mutual information under Markov assumption, followed by the three stepping algorithms LS1, LS2 and RC. We believe that this is caused by the range of the dataset is too large so that the battery size is too small for this dataset. Actually, if the capacity of battery is 1KWh, the result of BE is similar to that of LS1 and LS2. And if the capacity of battery is even larger, LS1 and LS2 outperform BE. Also, RC still dominates in the experiment of binary version mutual information.

Reducing the Battery Capacity. We note that when one uses large-enough batteries, then either BE or NILL would leak little information, because both algorithms could hold the external load constant for long period of time with a large battery. However, a large battery would be expensive, and it seems unlikely that many users would be willing to pay substantially for protection against privacy concerns caused by smart meters. Therefore, we argue that research on BLH should focus on the cases of using small batteries. In our experiments, we consider batteries of size ranging from 0.1KWh to 1.2 KWh. According to [29], a 0.6KWh battery costs around $100.

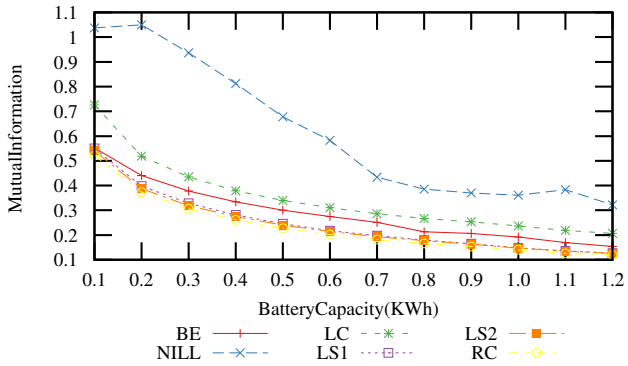
In this final set of experiments, we show that using stepping algorithms one can obtain good privacy protection with a smaller batter, and thus a lower cost. We use the two longest datasets M1 and S1. This requires searching through many difference battery parameters, and we use the mutual information measure under Markov assumption as the criterion of privacy level. Table 6 reports the comparison of such measurement among LS2, BE, and NILL. For the dataset S1, to gain similar privacy level, BE needs a battery with the capacity that is 1.41 times that of LS2, and NILL needs 3.54 times. On the M1 dataset, the ratio is even larger, 3.54 for BE and 6.62 for NILL.

7. RELATED WORK

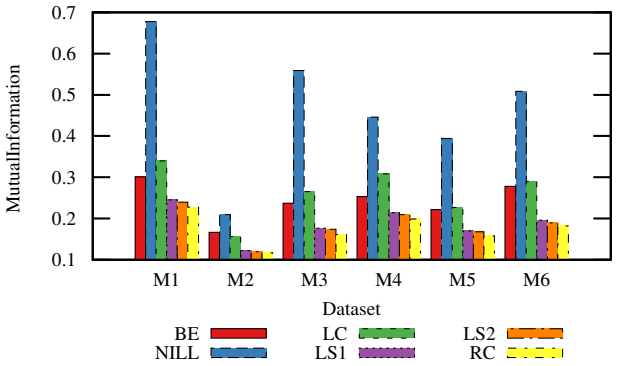
Methods have been proposed to extend existing NILM algorithms. In [22], Kim et al. propose a method to extend NILM algorithms to cases where one does not have a priori knowledge of appliance signatures using hidden Markov models. In the case where fairly exact appliance signatures are known, the steady state load can be disaggregated into its individual loads by solving a binary knapsack problem [26]. In [31], Molina-Markham et al. show how to use off-the-shelf statistical tools to detect household habits from power consumption patterns. In another study, Lisovich et al. [28] show that in a living environment monitored by cameras, sleep schedules and presence of occupants could be determined with over 90% accuracy after a three days training phase.

Varodayan and Khisti [42] propose another battery system in a simplified binary model where at each time the demand load and the external load can be either 0 or 1. They also use mutual information between the external load and the demand load to measure information leakage, and show that stochastic battery policies can decrease information leakage with respect to the best-effort algo-

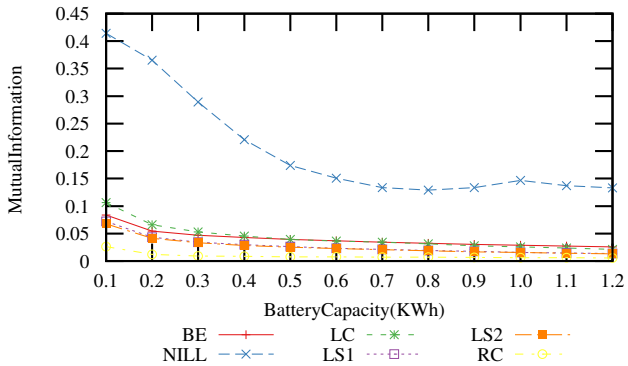
[2]A 0.5 KWh battery at 120 V nominal would retail from $100-$300 USD.



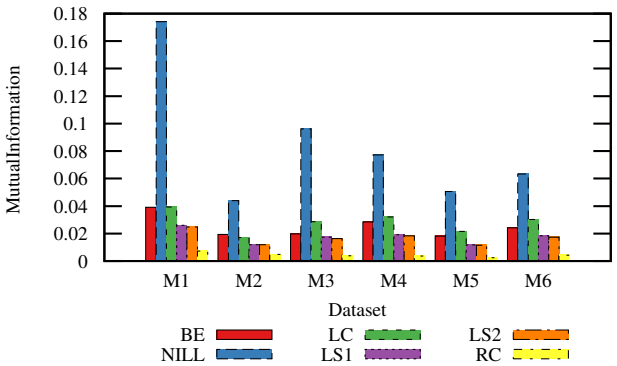
(a) MI between $e'(t)$ and $d'(t)$ under independence assumption. Varying battery capacity with dataset M1.



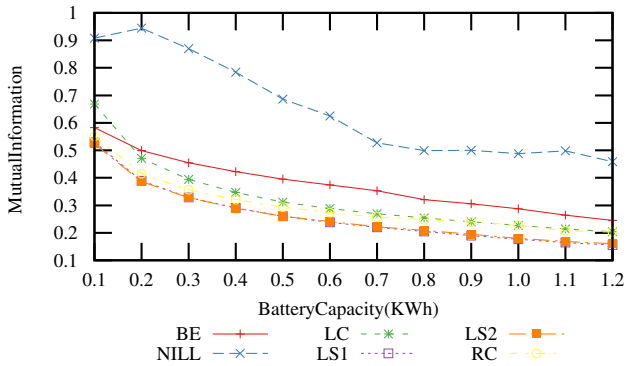
(b) MI between $e'(t)$ and $d'(t)$ under independence assumption. Varying datasets with battery set at 0.5KWh.



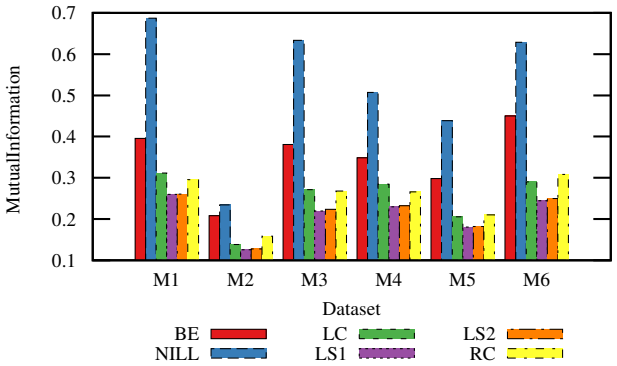
(c) MI between binary version of $e'(t)$ and $d'(t)$ under independence assumption. Varying battery capacity with dataset M1



(d) MI between binary version of $e'(t)$ and $d'(t)$ under independence assumption. Varying datasets, with battery set at 0.5KWh

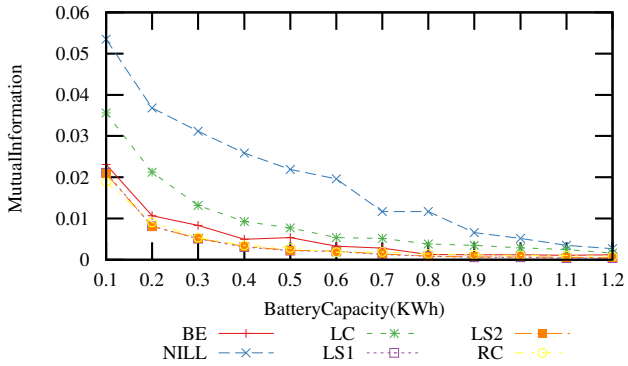


(e) MI between $e'(t)$ and $d'(t)$ under Markov assumption. Varying batter capacity with dataset M1.

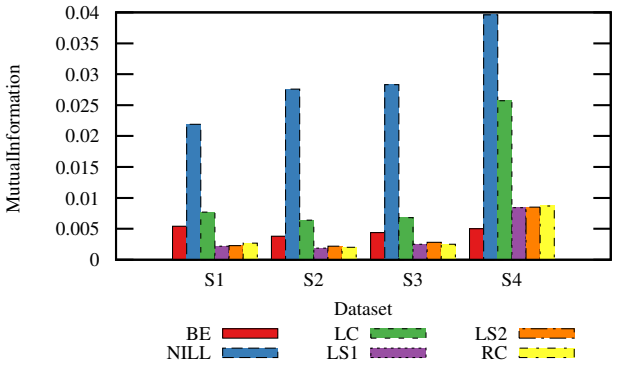


(f) MI between $e'(t)$ and $d'(t)$ under Markov assumption. Varying datasets with battery set at 0.5KWh.

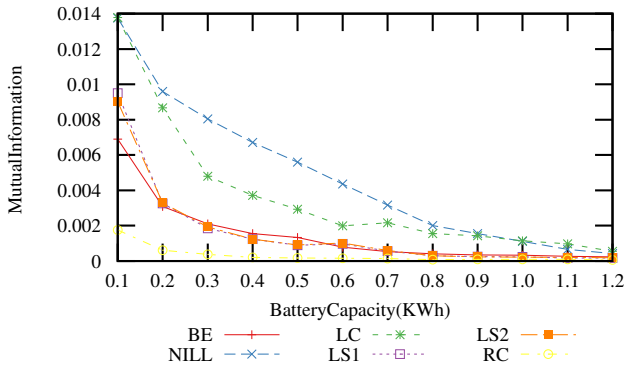
Figure 3: One-minute data with three mutual information, smaller values mean better privacy.



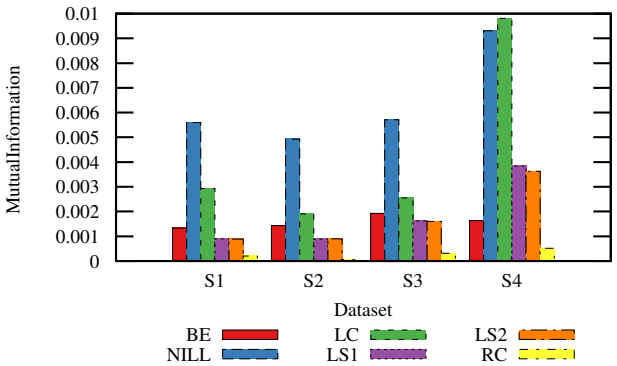
(a) MI between $e'(t)$ and $d'(t)$ under independence assumption. Varying battery capacity with dataset S1.



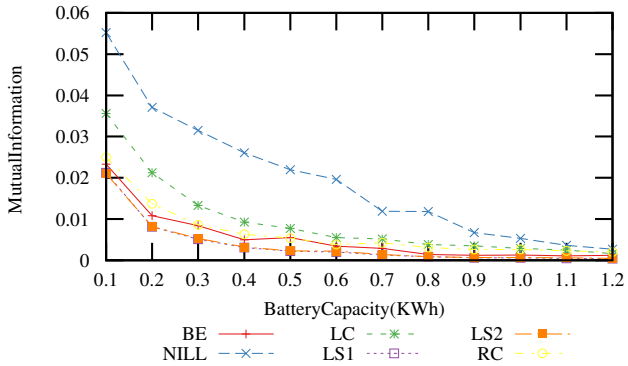
(b) MI between $e'(t)$ and $d'(t)$ under independence assumption. Varying datasets with battery set at 0.5KWh.



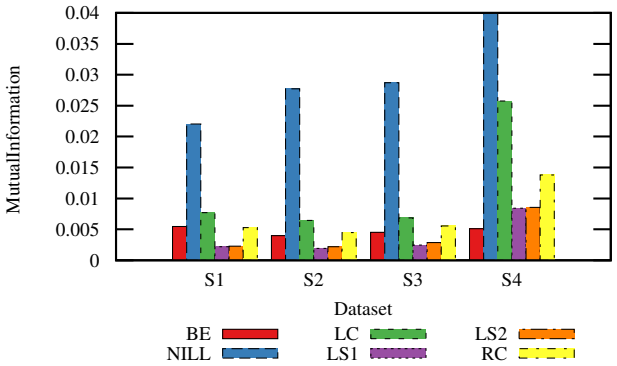
(c) MI between binary version of $e'(t)$ and $d'(t)$ under independence assumption. Varying battery capacity with dataset S1



(d) MI between binary version of $e'(t)$ and $d'(t)$ under independence assumption. Varying datasets, with battery set at 0.5KWh



(e) MI between $e'(t)$ and $d'(t)$ under Markov assumption. Varying batter capacity with dataset S1.



(f) MI between $e'(t)$ and $d'(t)$ under Markov assumption. Varying datasets with battery set at 0.5KWh.

Figure 4: One-second data with three mutual information, smaller values mean better privacy.

One-second dataset (Varying Battery)				One-second dataset (Varying Datasets)			
Battery	**DS**	**NILL**	**BE**	**Battery**	**DS**	**NILL**	**BE**
0.9KWh	S1	1942.9/2013.2(96.51%)	28.3/28.3(100.0%)	1.0KWh	S1	752.5/762.1(98.78%)	25.0/25.0(100.0%)
1.0KWh	S1	752.5/761.8(98.78%)	25.0/25.0(100.0%)	1.0KWh	S2	32.4/32.4(100.0%)	2.7/2.7(100.0%)
1.1KWh	S1	699.4/728.2(96.04%)	12.8/12.8(100.0%)	1.0KWh	S3	1101.4/1136.3(96.93%)	12.4/12.4(100.0%)
1.2KWh	S1	221.4/221.4(100.0%)	5.8/5.8(100.0%)	1.0KWh	S4	1080.2/1140.2(94.74%)	19.5/19.5(100.0%)
One-minute dataset (Varying Battery)				One-minute dataset (Varying Datasets)			
Battery	**DS**	**NILL**	**BE**	**Battery**	**DS**	**NILL**	**BE**
0.7KWh	M1	133.5/146.4(91.19%)	48.7/59.2(82.26%)	1.0KWh	M1	67.9/76.3(88.99%)	42/47.2(88.98%)
0.8KWh	M1	94.4/106.1(88.97%)	46.8/55(85.09%)	1.0KWh	M2	56.0/62.3(89.89%)	31.7/37.4(84.76%)
0.9KWh	M1	74.4/81.4(91.40%)	44.1/50.8(86.81%)	1.0KWh	M3	44.6/47.1(94.69%)	20.6/25.9(79.54%)
1.0KWh	M1	67.9/76.3(88.99%)	42/47.2(88.98%)	1.0KWh	M4	91.1/95.8(95.09%)	24.5/28.8(85.07%)
1.1KWh	M1	66.2/73.4(81.17%)	39.7/43.9(90.43%)	1.0KWh	M5	32.8/35.3(92.92%)	22.5/25.2(89.29%)
1.2KWh	M1	64.1/72.7(85.68%)	37.4/41.1(91.00%)	1.0KWh	M6	68.1/75.6(90.08%)	34.1/39.8(85.68%)

Table 5: Average number of load-change events detected per day. The DS column identifies the dataset used. The format is a/b(precision), where b is the number of load-change event detected, a is the number of accurate detection, precision $= a/b$. We say that a detection is accurate if the value of detected load-change is exactly the same as the value of the actual load-change. Note that the recall rate is determined by b, the number of detected events.

One-second dataset (S1)			One-minute dataset (M1)		
LS2	**BE**	**NILL**	**LS2**	**BE**	**NILL**
0.1KWh(0.0210)	0.1KWh(0.0233)	0.5KWh(0.0220)	0.1KWh(0.5233)	0.2KWh(0.4995)	0.7KWh(0.5269)
0.2KWh(0.0082)	0.3KWh(0.0084)	0.8KWh(0.0118)	0.2KWh(0.3860)	0.5KWh(0.3956)	1.7KWh(0.4110)
0.3KWh(0.0053)	0.5KWh(0.0055)	1.0KWh(0.0053)	0.3KWh(0.3280)	0.8KWh(0.3211)	2.0KWh(0.3323)
0.4KWh(0.0032)	0.6KWh(0.0034)	1.1KWh(0.0036)	0.4KWh(0.2899)	1.0KWh(0.2875)	2.3KWh(0.3009)
0.5KWh(0.0023)	0.7KWh(0.0029)	1.3KWh(0.0022)	0.5KWh(0.2607)	1.1KWh(0.2644)	2.6KWh(0.2668)
average ratio : 1	1.41	3.54	1	2.37	6.62

Table 6: To gain similar privacy, how large battery should LS2, BE, NILL use. The format of data is $capacity(mutual\_info)$.

rithm in this binary model. Rajagopalan et al. [37] also use the mutual information between the external load and the demand load in measuring privacy. No BLH algorithm was proposed in [37]. Acs et al. [3] add Laplace and truncated geometric noise to the external load through battery discharging in order to gain differential privacy guarantees, but do not consider the case where the battery must be recharged. Backes et al. [5] extends this scheme to include battery recharging, and shows that the power consumption of a television can be made differentially private with very large batteries.

A different approach to obtaining privacy guarantees for load profiles is to trust the utility to implement privacy protections in the meter. One of the first approaches suggested is to use a zero knowledge protocol between a third party and the utility to report load aggregates from the meter [32, 38, 19]. This allows for time of day billing to be done without releasing fine-grained load profiles. Differential privacy has also been proposed for load profiles [2] by adding Laplacian noise. However, this work does not apply to time-varying billing rates. In [41] Shi et al. use homomorphic encryption to make guarantees about sums taken at an aggregation point. Our paper is orthogonal to this line of research. For these cryptographic mechanisms to be effective, they need to be adopted by smart meter vendors and the utilities, and the smart meters and vendors must be trusted. As smart meters are already being deployed, it appears that the likely scenario is that the majority of deployed smart meters will not support these protocols. Our approach can be deployed with existing smart meters.

8. CONCLUSIONS

We have identified new vulnerabilities in two existing BLH algorithms that allow for the recovery of substantial appliance usage information. We have also introduced a novel stepping-based framework for BLH algorithms, which by design are secure against precise load change recovery attacks. We also propose mutual-information based measurements to evaluate the privacy of different algorithms. Experimental evaluation demonstrates the effectiveness of our approach, and in particular the LS2 stepping algorithm significantly and consistently outperforms other algorithms.

9. ACKNOWLEDGEMENTS

W. Yang, N. Li, and W. Qardaji were supported by the Air Force Office of Scientific Research MURI Grant FA9550-08-1-0265, and by the National Science Foundation under Grant No. 0905442. Y. Qi was supported by NSF IIS-0916443, NSF CAREER award IIS-1054903, and the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370. S. McLaughlin and P. McDaniel were partially supported by the National Science Foundation under Grant No. CCF 0937944 and CNS 0643907, and by a grant from the Security and Software Engineering Research Center (S^2ERC).

10. REFERENCES

[1] Autosense: A wireless sensor system to quantify personal exposures to psychosocial stress and addictive substances in natural environments. http://sites.google.com/site/autosenseproject.

[2] G. Acs and C. Castelluccia. I have a DREAM! (DiffeRentially privatE smArt Metering). In *13th Information Hiding Conference*, 2011.

[3] G. Acs, C. Castelluccia, and W. Lecat. Protecting against Physical Resource Monitoring. In *10th ACM Workshop on Privacy in the Electronic Society*, 2011.

[4] R. Anderson and S. Fuloria. On the security economics of electricity metering. In *Proceedings of the 9th Workshop on the Economics of Information Security (WEIS)*, 2010.

[5] M. Backes and S. Meiser. Differentially Private Smart Metering with Battery Recharging. iacr.org eprint, 2012.

[6] M. Baranski and J. Voss. Detecting patterns of appliances from total load data using a dynamic programming approach. *IEEE International Conference on Data Mining*, pages 327–330, 2004.

[7] M. Baranski and J. Voss. Genetic algorithm for pattern detection in nialm systems. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2004.

[8] M. Berges, E. Goldman, H. S. Matthews, and L. Soibelman. Training load monitoring algorithms on highly sub-metered home electricity consumption data. *Tsinghua Science & Technology*, 13(Supplement 1):406–411, 2008.

[9] D. Bergman, D. Jin, J. Juen, N. Tanaka, C. Gunter, and A. Wright. Nonintrusive Load-Shed Verification. *Pervasive Computing, IEEE*, 10(1):49–57, jan.-march 2011.

[10] A. Brothman, R. D. Reiser, N. L. Kahn, F. S. Ritenhouse, and R. A. Wells. Automatic remote reading of residential meters. *IEEE Transactions on Communication Technology*, 13(2):219 – 232, 1965.

[11] W. L. Chan, A. T. P. So, and L. L. Lai. Harmonics load signature recognition by wavelets transforms. In *Proceedings of the International Conference on Electric Utility Deregulation and Restructuring and Power Technologies*, 2000.

[12] L. Farinaccio and R. Zmeureanu. Using a pattern recognition approach to disaggregate the total electricity consumption in a house into the major end-uses. *Energy and Buildings*, 30(3):245–259, 1999.

[13] K. Fehrenbacher. Smart meter worm could spread like a virus. `http://earth2tech.com/2009/07/31/smart-meter-worm-could-spread-like-a-virus/`.

[14] M. Goldberg. Measure twice, cut once. *IEEE Power and Energy Magazine*, pages 46 – 54, May/June 2010.

[15] M. E. Guedri, G. D'Urso, C. Lajaunie, and G. Fleury. Time-Frequency Characterisation for Electric Load Monitoring. In *Proceedings of the 17th European Signal Processing Conference (EUSIPCO)*, 2009.

[16] G. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870 –1891, dec 1992.

[17] G. W. Hart. Residential energy monitoring and computerized surveillance via utility power flows. *IEEE Technology and Society Magazine*, June 1989.

[18] J. Healey and R. Picard. Detecting Stress During Real-World Driving Tasks Using Physiological Sensors. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):156–166, 2005.

[19] M. Jawurek, M. Johns, and F. Kerschbaum. Plug-in Privacy for the Smart Grid. In *11th Privacy Enhacning Technologies Symposium*, 2011.

[20] G. Kalogridis, C. Efthymiou, S. Denic, T. Lewis, and R. Cepeda. Privacy for smart meters: Towards undetectable appliance load signatures. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 232 –237, oct. 2010.

[21] R. Kelley and R. D. Pate. Mesh Networks and Outage Management. White Paper, September 2008.

[22] H. Kim, M. Marwah, M. F. Arlitt, G. Lyon, and J. Han. Unsupervised disaggregation of low frequency power measurements. In *SDM*, pages 747–758. SIAM / Omnipress, 2011.

[23] C. S. King. The Economics of Real-Time and Time-of-Use Pricing For Residential Consumers. Technical report, American Energy Institute, 2001.

[24] B. Krebs. Experts: Smart grid poses privacy risks, 2009.

[25] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong. Power Signature Analysis. *Power and Energy Magazine, IEEE*, 1(2):56–63, Mar-Apr 2003.

[26] M. LeMay, J. J. Haas, and C. A. Gunter. Collaborative recommender systems for building automation. *Hawaii International Conference on System Sciences*, 0:1–10, 2009.

[27] A. Leo. The Measure of Power. *Technology Review Magazine*, June 2001.

[28] M. A. Lisovich, D. K. Mulligan, and S. B. Wicker. Inferring personal information from demand-response systems. *IEEE Security and Privacy*, 8(1):11–20, 2010.

[29] 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*, CCS '11, pages 87–98, New York, NY, USA, 2011. ACM.

[30] S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, and P. McDaniel. Multi-vendor Penetration Testing in the Advanced Metering Infrastructure. In *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC)*, 2010.

[31] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 61–66, New York, NY, USA, 2010. ACM.

[32] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (BuildSys)*, 2010.

[33] H. Murata and T. Onoda. Applying kernel based subspace classification to a non-intrusive monitoring for household electric appliances. In *Proceedings of the 11th International Conference on Artificial Neural Networks*, 2001.

[34] 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 (nominated for the best paper award). In *Ubicomp*, pages 271–288, 2007.

[35] Privacy by Design. Smartprivacy for the smart grid. `http://www.futureofprivacy.org/`, 2009.

[36] E. L. Quinn. Smart metering and privacy: Existing law and competing policies. A report for the Colorado Public Utilities Commission, 2009.

[37] S. Rajagopalan, L. Sankar, S. Mohajer, and H. Poor. Smart meter privacy: A utility-privacy framework. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 190 –195, oct. 2011.

[38] A. Rial and G. Danezis. Privacy-Preserving Smart Metering. Technical Report MSR-TR-2010-150, Microsoft Research, November 2010.

[39] I. Richardson and M. Thomson. One-minute resolution domestic electricity use data, 2008-2009 [computer file], Oct. 2010. Colchester, Essex: UK Data Archive [distributor], SN: 6583, http://dx.doi.org/10.5072/UKDA-SN-6583-1.

[40] J. Roos, I. Lane, E. Botha, and G. Hancke. Using neural networks for non-intrusive monitoring of industrial electrical loads. In *Proceedings of the 10th Instrumentation and Measurement Technology Conference (IMTC '94)*, 1994.

[41] E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *18th Network and Distributed Systems Security Symposium*, 2011.

[42] D. Varodayan and A. Khisti. Smart meter privacy using a rechargeable battery: Minimizing the rate of information leakage. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1932 –1935, may 2011.

[43] G. P. Zachary. Saving Smart Meters From a Bakclash. *IEEE Spectrum*, 2011.

APPENDIX

A. MUTUAL INFORMATION MEASURES

Mutual information between $\mathbf{e}'$ and $\mathbf{d}'$ allows us to quantitatively measure how much common information embedded in $\mathbf{e}'$ and $\mathbf{d}'$, regardless what algorithms—linear or nonlinear, deterministic or stochastic—have been used to generate $\mathbf{e}'$ given $\mathbf{d}'$.

Mutual Information Under the Independence Assumption. Using the independence assumption, we can easily compute the mutual information $\mathcal{I}$ as follows. We first discretize $e'(t)$ (and $d'(t)$) into K discrete values (in our experiments, we set $K = 500$).

We then estimate the joint distribution $p(f(t))$, where $f(t) = (e'(t), d'(t)$ at each time by simply counting the number of the joint appearance of (a, b) and normalizing it:

$$p(e'(t) = a, d'(t) = b) = \frac{\sum_{i=1}^{T} \delta(e'(i) = a \wedge e'(i) = b)}{T}$$

where $\delta(\cdot)$ is 1 if the statement inside is true. Given the joint distribution, we can easily obtain the marginal distributions of $e'(t)$ and $d'(t)$: where $p(e'(t)) = \sum_{d'(t)} p(e'(t), d'(t))$ and $p(d'(t)) = \sum_{e'(t)} p(e'(t), d'(t))$.

Given the joint and marginal distributions, we calculate the mutual information as follows:

$$\begin{aligned}\mathcal{I}(\mathbf{e}'||\mathbf{d}') &= \sum_{i=1,\ldots,T} \mathcal{I}(e'(i)||d'(i)) \\ &= \sum_{i} \sum_{e'(i)} \sum_{d'(i)} p(e'(i), d'(i)) \log \frac{p(e'(i), d'(i))}{p(e'(i))p(d'(i))}\end{aligned} \quad (1)$$

Mutual Information Under the Markov Assumption. While the independence assumption makes the computation very efficient, it ignores correlations embedded in samples of $e'(t)$ (and of $d'(t)$)—as time series, these samples are naturally correlated (e.g., $e'(t)$ may depend on $e'(t-1)$). To address this issue, we model samples in $f(t)$, $e'(t)$ and $d'(t)$ by stationary first-order Markov chains. As a result, we have

$$p(e'(t) = a) = \frac{\sum_{i=1}^{T} \delta(e'(i) = a)}{T}$$

$$p(e'(t) = a | e'(t-1) = c) = \frac{\sum_{i=2}^{T} \delta(e'(i) = a \wedge e'(i-1) = c)}{\sum_{i=2}^{T} \delta(e'(i-1) = c)}$$

$$p([e'(1), \ldots, e'(T)]) = p(e'(1)) \prod_{i=2}^{T} p(e'(i) | e'(i-1))$$

Similarly we can compute the distribution of $([d'(1), \ldots, d'(T)])$. To obtain the joint distribution over $(\mathbf{e}', \mathbf{d}')$, we compute

$$\begin{aligned}&p(f(t) = (a, b) \mid f(t-1) = (c, d)) \\ &\qquad = \frac{\sum_{i=2}^{T} \delta(f(i) = (a, b) \wedge f(i-1) = (c, d))}{\sum_{i=2}^{T} \delta(f(i-1) = (c, d))}, \\ &p(\mathbf{e}', \mathbf{d}') = p(f(1)) \prod_{i=2}^{T} p(f(i) | f(i-1)).\end{aligned}$$

Then the mutual information is

$$\begin{aligned}\mathcal{I}(\mathbf{e}'||\mathbf{d}') &= \sum_{\mathbf{e}'} \sum_{\mathbf{d}'} p(\mathbf{e}', \mathbf{d}') \log \frac{p(\mathbf{e}', \mathbf{d}')}{p(\mathbf{e}')p(\mathbf{d}')} \\ &= \sum_{i=1}^{T-1} \mathcal{I}(e'(i, i+1)||d'(i, i+1)) - \\ &\quad \sum_{i=2}^{T-1} \mathcal{I}(e'(i)||d'(i))\end{aligned} \quad (2)$$

where $\mathcal{I}(e'(i, i+1)||d'(i, i+1))$

$$= \sum_{f(i)} \sum_{f(i+1)} p(f(i-1))p(f(i)|f(i-1)) \cdot \log \frac{p(f(i-1))p(f(i)|f(i-1))}{p(e'(i-1, i))p(d'(i-1, i))}$$

and $\mathcal{I}(e'(i)||d'(i))$ is given in Equation (1).