I have a DREAM!
(DIffeRentially PrivatE smart Metering)

Technical report

March 15, 2011

Abstract

This paper presents a new privacy-preserving smart metering system. Our scheme is private under the differential privacy model and therefore provides strong and provable guarantees. With our scheme, an (electricity) supplier can periodically collect data from smart meters and derive aggregated statistics without learning anything about the activities of individual households. For example, a supplier cannot tell from a user's trace whether or when he watched TV or turned on heating. Our scheme is simple, efficient and practical. Processing cost is very limited: smart meters only have to add noise to their data and encrypt the results with an efficient stream cipher.

Contents

1 Introduction

Several countries throughout the world are planning to deploy smart meters in households in the very near future. The main motivation, for governments and electricity suppliers, is to be able to match consumption with generation.

Traditional electrical meters only measure total consumption on a given period of time (i.e., one month or one year). As such, they do not provide accurate information of when the energy was consumed. Smart meters, instead, monitor and report consumption in intervals of few minutes. They allow the utility provider to monitor, almost in real-time, consumption and possibly adjust generation and prices according to the demand. Billing customers by how much is consumed and at what time of day will probably change consumption habits to help matching consumption with generation. In the longer term, with the advent of smart appliances, it is expected that the smart grid will remotely control selected appliances to reduce demand.

Problem Statement: Although smart metering might help improving energy management, it creates many new privacy problems [1]. Smart meters provide very accurate consumption data to electricity providers. As the interval of data collected by smart meters decreases, the ability to disaggregate low-resolution data increases. Analyzing high-resolution consumption data, Nonintrusive Appliance Load Monitoring (NALM) [14] can be used to identify a remarkable number of electric appliances (e.g., water heaters, well pumps, furnace blowers, refrigerators, and air conditioners) employing exhaustive appliance signature libraries. Researchers are now focusing on the myriad of small electric devices around the home such as personal computers, laser printers, and light bulbs [18]. Moreover, it has also been shown that even simple off-the-shelf statistical tools can be used to extract complex usage patterns from high-resolution consumption data [19]. This extracted information can be used to profile and monitor users for various purposes, creating serious privacy risks and concerns. As data recorded by smart meters is lowering in resolution, and inductive algorithms are quickly improving, it is urgent to develop privacy-preserving smart metering systems that provide strong and provable guarantees.

Contributions: We propose a privacy-preserving smart metering scheme that guarantees users' privacy while still preserving the benefits and promises of smart metering. Our contributions are many-fold and summarized as follows:

- We provide the first provably private and distributed solution for smart metering that optimizes utility without relying on a third trusted party (i.e., an aggregator). We were able to avoid the use of a third trusted party by proposing a new distributed Laplacian Perturbation Algorithm (DLPA).

  In our scheme, smart meters are grouped into clusters, where a cluster is a group of hundreds or thousands of smart meters corresponding, for example, to a quarter of a city. Each smart meter sends, at each sampling period, their measures to the supplier. These measures are noised and encrypted such that the supplier can compute the noised aggregated electricity consumption of the cluster, at each sampling period, without getting access to individual values. The aggregate is noised just enough to provide differential privacy to each participating user, while still providing high utility (i.e., the low error). Our scheme is secure under the differential privacy model and

therefore provides strong and provable privacy guarantees. In particular, we guarantee that the supplier cannot retrieve any information about any user consumption, whatever auxiliary information it knows about that user. Our scheme is simple, efficient and practical. It requires either one or two rounds of message exchanges between a meter and the supplier. Furthermore, processing cost is very limited: smart meters only have to add noise to their data and encrypt the results with an efficient stream cipher. Finally, our scheme is robust against smart meter failures and malicious nodes. More specifically, it is secure even if α fraction of all nodes of a cluster collude with the supplier, where α is a security parameter.

- We provide a detailed analysis of the security and performance of our proposal. The security analysis is performed analytically. The performance, which is evaluated using the utility metric, is performed using simulation. We implemented a new electricity trace generation tool based on [22] which generates one-minute resolution synthetic consumption data of different households.

2 Related Work

Several papers addressed the privacy problems of smart meters in the recent past [9, 19, 1, 2, 3, 21, 15, 12]. However, only a few of them have proposed technical solutions to protect users' privacy. In [1, 2], the authors discuss the different security aspects of smart metering and the conflicting interests among stakeholders. The privacy of billing is considered in [21, 19]. These techniques uses zero-knowledge proofs to ensure that the fee calculated by the user is correct without disclosing any consumption data.

Seemingly, the privacy of monitoring the sum consumption of multiple users may be solved by simply anonymizing individual measurements like in [9] or using some mixnet. However, these "ad-hoc" techniques are dangerous and do not provide any real assurances of privacy. Several prominent examples in the history have shown that ad-hoc methods do not work [16]. Moreover, these techniques require an existing trusted third party who performs anonymization. The authors in [3] also perturb the released aggregate with random noise and use a different model from ours to analyze the privacy of their scheme. However, they do not encrypt individual measurements which means that the added noise must be large enough to guarantee reasonable privacy. As individual noise shares sums up at the aggregation, the final noise makes the aggregate useless. In contrast to this, [12] uses homomorphic encryption to guarantee privacy for individual measurements. However, the aggregate is not perturbed which means that it is not differential private.

The notion of differential privacy was first proposed in [8]. Differential privacy says that releasing data using a differentially private algorithm will not increase the adversary's chance to infer any information about any users in the dataset. The main advantage of differential privacy over other privacy models is that it does not specify the prior knowledge of the adversary. Initial works on differential privacy focused on the problem that how a trusted curator (aggregator), who collects all data from users, can differential privately release statistics. By contrast, our scheme ensures differential privacy even if the curator is untrusted. Although [7] described protocols for generating shares of random noise which is secure against malicious participants, it requires communication between users and it uses expensive secret sharing techniques resulting in high overhead in case of large number of users. Similarly, traditional Secure Multiparty Computation (SMC) techniques [13] [5] also require interactions between

users. All these solutions are impractical for resource constrained smart meters where all the computation is done by the aggregator and users are not supposed to communicate with each other.

Two closely related works to ours are [20] and [23]. In [20], the authors propose a scheme to differential privately aggregate sums over multiple slots when the aggregator is untrusted. However, they use the threshold Paillier cryptosystem [11] for homomorphic encryption which is much more expensive compared to [4] that we use. They also use different noise distribution technique which requires several rounds of message exchanges between the users and the aggregator. By contrast, our solution is much more efficient and simple: it requires only a single message exchange if there are no node failures, otherwise, we need only one extra round. In addition, our solution does not rely on expensive public key cryptography during aggregation.

A recent paper [23] proposes another technique to privately aggregate time series data. They use a different noise generation method from [20] and ours, but their scheme only satisfies the relaxed (ε, δ)-differential privacy definition. Indeed, in their scheme, each node adds noise probabilistically which means that none of the nodes add noise with some positive probability δ. If this happens, there is a privacy breach. Although δ can be arbitrarily small, this also decreases the utility. By contrast, in our scheme, $\delta = 0$ while ensuring nearly optimal utility. In addition, [23] is not robust against node and communication failures.

3 The model

3.1 Network model

The network is composed of four major parts: the *supplier/aggregator*, the *electricty distribution network*, the *communication network*, and the *users* (customers). Every user is equipped with an electricity smart meter, which has two modes of operation: (1) in emergency mode, if the consumption is above a certain threshold, which likely indicates some abnormal event or failure, data is sent directly to the supplier. In this case, we cannot guarantee privacy as the supplier needs to know the location of the failure to fix it. (2) In the operational mode, each smart meter measures the electricity consumption of the user in every slot with length T_s, and, using the communication network, sends the measurement to the aggregator at the end of every slot (in practice, T_s is around 1-30 minutes). Note that the communication and distribution network can be identical (e.g., if we use some PLC technology to transfer data). The motivation behind monitoring the sum consumption of different users is to improve resource management and adjust generation and prices according to the demand.

The measurement of user i in slot t is denoted by X_t^i. The consumption profile of user i is described by the vector $(X_1^i, X_2^i, \ldots)$. Privacy directly correlates with T_s; finer-grained samples means more accurate profile, but also entails weaker privacy. The supplier is interested in the sum of all measurements in every slot (i.e., $\sum_{i=1}^{N} X_t^i \stackrel{\mathsf{def}}{=} \mathbf{X}_t$). The supplier can either obtain them from the meters directly, or from a separate aggregator entity who though retrieves them from the meters, performs the aggregation, and forwards the aggregate to the supplier.

Like in [3], we also assume that smart meters are trusted devices (i.e., tamper-resistant) which can store key materials and perform crypto computations. This realistic assumption has also been confirmed in [2]. We further suppose that the supplier knows the average daily

consumption of each user. [1]

We assume that each node is configured with a private key and gets the corresponding certificate from a trusted third party. For example, each country might have a third party that generates these certificate and can additionally generate the "supplier" certificates to supplier companies [2]. As in [2], we also assume that public key operations are employed only for initial key establishment, probably when a meter is taken over by a new supplier, and after that, all communication is protected by symmetric crypto-based techniques. Messages exchanged between the supplier and the meters are authenticated using pairwise MACs.

Smart meters are assumed to have bidirectional communication channel (using some wireless or PLC technology) with the supplier and/or the aggregator, but the meters cannot communicate with each other. Finally, we assume that nodes may (randomly) fail, and in these cases, cannot send their measurements to the aggregator/supplier. This can be caused by device or communication failures. However, nodes are supposed to use some reliable transport protocol to overcome the transient failures of the channel.

Finally, we note that smart meters also allow the supplier to perform fine-grained billing based on time-dependant variable tariffs. Here, we are not concerned with the privacy and security problems of this service. Interested readers are referred to [21, 19].

3.2 Adversary model

In general, the objective of the adversary is to infer detailed information about household activity (e.g, how many people are in home and what they are doing at a given time). In order to do that, it needs to extract complex usage patterns of appliances which includes the level of power consumption, its periodicity, and its duration. It has been shown in [19] that different data mining techniques can be easily applied to a raw consumption profile to obtain this information.

In terms of its capability, we distinguish three types of adversary. The first is the a *honest-but-curious (HC) adversary*, who attempts to obtain private information about a user, but it follows the protocol faithfully and do not provide false information [19]. It only uses the (non-manipulated) collected data.

The *dishonest-but-non-intrusive (DN) adversary* may not follow the protocol correctly and is allowed to provide false information to manipulate the collected data. Some users can also be malicious and collude even with the supplier to collect information about honest users. However, the DN adversary is not allowed to access and modify the distribution network to mount attacks. For instance, it cannot install any extra hardware in the distribution network to collect more information about users.

In addition to the DN adversary, the strongest *dishonest-and-intrusive (DI) adversary* can invade the distribution network and modify it to gather more information about users. It can access the network infrastructure and, e.g., monitor the consumption on every power line outside the users' households by installing extra meters.

For all cases, we suppose that the adversary can have any kind of extra knowledge about honest users, beyond the legitimately collected data, which might help to infer private infor-

[1] e.g., at the end of every longer period with length T_p, where $T_s \ll T_p$ and T_p is around a few month, each meter sends the real consumption aggregated over the last T_p/T_s slots directly to supplier. This assumption do not entail more privacy risks than the old-fashioned traditional metering technique used nowadays.

mation about them. For instance, it can observe their daily habits and their other activities[2], or obtain extra information by doing personal interviews, surveys, etc.

3.3 Privacy model

We use differential privacy [8] that models the adversary described above. In particular, differential privacy guarantees that a user's privacy should not be threatened substantially more if he provides his measurement to the supplier regardless what external knowledge the supplier has. Intuitively, a sanitization algorithm is differential private, if any single user's data changes the distribution of its output insignificantly, and hence, the output practically does not leak any information about any user's data.

Definition 1 (ε-differential privacy) *An algorithm $\mathcal{A}$ is ε-differential private, if for all data sets D_1 and D_2, where D_1 and D_2 differ in at most a single user, and for all subsets of possible answers $S \subseteq Range(\mathcal{A})$,*

$$P(\mathcal{A}(D_1) \in S) \leq e^{\varepsilon} \cdot P(\mathcal{A}(D_2) \in S)$$

Differential private algorithms produce indistinguishable outputs for similar inputs (more precisely, differing by a single entry), and thus, the modification of any single user's data in the dataset (including its removal or addition) changes the probability of any output only up to a multiplicative factor e^{ε}. The parameter ε allows us to control the level of privacy. Lower values of ε implies stronger privacy, as they restrict further the influence of a user's data on the output. Note that differential privacy does not make any assumptions about the adversary's computational power and its external knowledge about the dataset. For instance, this model guarantees privacy for a user even if all other users' data is known to the adversary (e.g., it knows all measurements comprising the aggregate except the target user's) independently of what extra knowledge it has about the user beyond the sanitized data. In our adversary model, this is equivalent to the case when $N-1$ out of N users are malicious and cooperate with the supplier.

The definition of differential privacy also maintains a *composability property*: the composition of differential private algorithms remains differential private and their ε parameters are accumulated. In particular, a protocol having t rounds, where each round is individually ε differential private, is itself $t \cdot \varepsilon$ differential private.

4 Objectives

Our goal is to develop a practical scheme that preferably does not introduce more privacy risks for users than traditional metering systems while retaining the benefits of smart meters. More specifically, the scheme should be

- *differentially private*: Considering **DN adversary**, the scheme differential privately releases sanitized aggregates $\hat{\mathbf{X}}_t$ where the leaked private information about users is measured by ε.
- *robust and easily configurable*: It tolerates (random) node failures.

[2]Similarly to monitoring neighbors. Indeed, neighbors can also be malicious users, which is included in our model.

- *efficient*: It has low overhead which includes low computation load on smart meters, and low communication overhead between the supplier and individual meters. It should use pubic key operations only for initial key establishment. Afterwards, all communication is protected using more efficient symmetric crypto-based techniques.

- *distributed*: Besides a certificate authority, the protocol does not require any trusted third party such as a trusted aggregator as in [3]. The smart meters communicate directly with the supplier.

- *useful for the supplier*: The sanitized and the original (non-sanitized) aggregate should be "similar". This sanitized data should also be sufficient for the supplier to accomplish its desired goals. For instance, this includes the supplier's ability to perform efficient management of the resource: to monitor the consumption at the granularity of a maximum few hundred households, and to detect consumption peaks or abnormal consumption.

5 Building blocks

5.1 Output perturbation: achieving differential privacy

Let's say, we want to release the value of a function f applied on our dataset in a differential private way. The following theorem says that we can achieve differential privacy by releasing the perturbed output of f; simply adding a random noise to the value of f before releasing that, where the noise distribution is carefully calibrated to the global sensitivity of f, results in ε-differential privacy. The global sensitivity of a function is the maximum "change" in the value of the function when its input differs in a single entry. For instance, if f is simply the sum of all measurements, then the sensitivity is the maximum value that a measurement can have.

Theorem 1 ([8]) *For all $f : \mathbb{D} \to \mathbb{R}^r$, the following mechanism $\mathcal{A}$ is ε-differential private: $\mathcal{A}(D) = f(D) + \mathcal{L}(S(f)/\varepsilon)$, where $\mathcal{L}(S(f)/\varepsilon)$ is an independently generated random variable following the Laplace distribution and $S(f)$ denotes the global sensitivity of f*[3].

In particular, if each time the noise is drawn independently from the same Laplace distribution with PDF $P(\mathcal{L}(\lambda) = x) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}}$, we obtain that $\frac{P(\mathcal{A}(D_1)=z)}{P(\mathcal{A}(D_2)=z)} = \frac{P(\mathcal{L}(\lambda)=z-f(D_1))}{P(\mathcal{L}(\lambda)=z-f(D_2))} \leq e^{\frac{-||f(D_1)||_1+||f(D_2)||_1}{\lambda}} \leq e^{||f(D_1)-f(D_2)||_1/\lambda} \leq e^{S(f)/\lambda}$ for all z. This perturbation method is also called as LPA (Laplacian Perturbation Algorithm).

ε-differential privacy guarantees a *global* ε bound to all users, as the noise is calibrated to the *global* sensitivity $S(f)$. In the sequel, we distinguish an individual's indistinguishability as follows. Let D_1 and D_2 differ only in user U's data. Then, the upper bound $e^{||f(D_1)-f(D_2)||_1/\lambda}$ on $\frac{P(\mathcal{A}(D_1)\in S)}{P(\mathcal{A}(D_2)\in S)}$ (for all D_1, D_2, and S) represents U's indistinguishability, where $\varepsilon = ||f(D_1) - f(D_2)||_1/\lambda$ is U's *individual privacy bound*. In the rest, if it is ambiguous in the given context, we distinguish global ε bound from an individual's ε bound. As $e^{||f(D_1)-f(D_2)||_1/\lambda} \leq e^{S(f)/\lambda}$, the maximum of the individual bounds in the dataset is the global ε bound.

[3]Formally, let $f : \mathbb{D} \to \mathbb{R}^r$, then the global sensitivity of f is $S(f) = \max ||f(D_1) - f(D_2)||_1$, where D_1 and D_2 differ in a single entry and $||\cdot||_1$ denotes the L_1 distance.

Example 1 *To illustrate these definitions, consider a mini smart metering application, where users U_1, U_2, and U_3 need to send the sum of their measurements in three consecutive slots. The measurements of U_1, U_2 and U_3 are $(X_1^1 = 100, X_2^1 = 300, X_3^1 = 200)$, $(X_1^2 = 250, X_2^2 = 400, X_3^2 = 350)$, and $(X_1^3 = 50, X_2^3 = 150, X_3^3 = 200)$, resp. The nodes want differential privacy for the released sum with a global $\varepsilon = 0.5$ bound. Based on Theorem 1, they need to add $\mathcal{L}(\lambda = \sum_t \max_i X_t^i/0.5 = 2000)$ noise to the released sum in* ***each*** *slot. The individual ε bound of U_1 is $\sum_t X_t^1/\lambda = 0.3$, and similarly, 0.5 of U_2, and 0.2 of U_3. Another interpretation is that U_1 has $\varepsilon_1 = X_1^1/\lambda = 0.05$, $\varepsilon_2 = X_2^1/\lambda = 0.15$, and $\varepsilon_3 = X_3^1/\lambda = 0.1$ privacy in each individual slot, and $\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3 = 0.3$ considering all three slots following from the composition property of differential privacy.*

5.1.1 Utility definition

Let $f : \mathbb{D} \rightarrow \mathbb{R}$. In order to measure the utility, we quantify the difference between $f(D)$ and its perturbed value (i.e., $\hat{f}(D) = f(D) + \mathcal{L}(\lambda)$) which is the error introduced by LPA. A common scale-dependant error measure is the Mean Absolute Error (MAE), which is $\mathbb{E}|f(D) - \hat{f}(D)|$ in our case. However, the error should be dependent on the non-perturbed value of $f(D)$; if $f(D)$ is greater, the added noise becomes small compared to $f(D)$ which intuitively results in better utility. Hence, we rather use a slightly modified version of a scale-independent metric called Mean Absolute Percentage Error (MAPE), which shows the proportion of the error to the data, as follows.

Definition 2 (Error function) *Let $D_t \in \mathbb{D}$ denote a dataset in time-slot t. Furthermore, let $\delta_t = \frac{|f(D_t) - \hat{f}(D_t)|}{f(D_t)+1}$ (i.e., the value of the error in slot t). The error function is defined as $\mu(t) = \mathbb{E}(\delta_t)$. The expectation is taken on the randomness of $\hat{f}(D_t)$. The standard deviation of the error is $\sigma(t) = \sqrt{Var(\delta_t)}$ in time t.*

Compared to the original definition of MAPE (i.e., $\mathbb{E}(|f(D_t) - \hat{f}(D_t)|/f(D_t))$) we add 1 to the denominator in order to avoid the case when it is zero and to bound the error value. Note that if $f(D_t) \gg 1$, then there is no significant difference between the value of $\mu(t)$ and MAPE, which is the case for most practical applications.

Example 2 *Continuing Example 1, $f(D_1) = \sum_i X_1^i + \mathcal{L}(\lambda)$, where $\lambda = 2000$. Moreover, $\mu(1) = \mathbb{E}\left(\frac{|f(D_1) - f(D_1) - \mathcal{L}(\lambda)|}{f(D_1)+1}\right) = \frac{\mathbb{E}|\mathcal{L}(\lambda)|}{f(D_1)+1} = \frac{\lambda}{f(D_1)+1}$ which is about 4.98 in the specific example. Similarly, $\mu(2) = 2000/851 \approx 2.35$ and $\mu(3) \approx 2.66$ (i.e., the released sums are practically useless). In addition, $\sigma(t) = \frac{\sqrt{Var|\mathcal{L}(\lambda)|}}{f(D_1)+1} = \frac{\sqrt{\mathbb{E}|\mathcal{L}(\lambda)|^2 - (\mathbb{E}|\mathcal{L}(\lambda)|)^2}}{f(D_1)+1} = \frac{\lambda}{f(D_1)+1} = \mu(t)$. In order to increase utility, we can decrease privacy. For instance, suppose we want to guarantee a global $\varepsilon = 0.5$ bound in individual slots instead of during three slots. This means that we need to add noise $\lambda = \max_i X_t^i/0.5$ in slot t (instead of $\sum_t \max_i X_t^i/0.5$) which means that $\mu(1), \mu(2)$, $\mu(3)$ change to 0.62, 0.47, and 0.53, resp. Although these are still not satisfactory, we can further decrease the error by taking the sum of more measurements (i.e., including more than 3 users into the aggregation).*

5.2 Additively homomorphic encryption

A homomorphic encryption scheme allows arithmetic operations to be performed on ciphertexts. They are especially useful in scenarios where someone who does not have decryption keys needs to perform arithmetic operations on a set of ciphertexts. Let $Enc()$ denote a

probabilistic encryption scheme. Let M be the message space and C the ciphertext space such that both of them are groups under the addition operation. $Enc()$ is an additive-homomorphic encryption scheme if for any instance $Enc()$ of the encryption scheme, given $c_1 = Enc_{k_1}(m_1)$ and $c_2 = Enc_{k_2}(m_2)$, there exists a key k such that

$$c_1 + c_2 = Enc_k(m_1 + m_2)$$

In other words, the result of adding plaintext values may be obtained by decrypting the sum of the corresponding encrypted values. A simple and practical *additively homomorphic* encryption technique, which suits perfectly for privacy-preserving additive aggregation, is the CaTsMy scheme [4]:

Encryption: Let k be a randomly generated keystream, where $k \in [0, \Delta - 1]$. After representing message m as integer $m \in [0, \Delta - 1]$ where Δ is a large integer, compute $c = Enc(m, k, \Delta) = m + k \pmod{\Delta}$

Decryption: $Dec(c, k, \Delta) = c - k \pmod{\Delta}$

Addition of ciphertexts: Let $c_1 = Enc(m_1, k_1, \Delta)$ and $c_2 = Enc(m_2, k_2, \Delta)$. For $k = k_1 + k_2$, $Dec(c_1 + c_2, k, \Delta) = m_1 + m_2$.

This scheme uses modular addition $(+)$ and ensures if $c_1 = Enc(m_1, k_1, \Delta)$ and $c_2 = Enc(m_2, k_2, \Delta)$ then $c_1 + c_2 = Enc(m_1 + m_2, k_1 + k_2, \Delta)$. If n different ciphertexts are added and $p = \max(m_i)$ then Δ should be selected as $\Delta = 2^{\lceil \log_2(p \cdot n) \rceil}$.

6 Overview of approaches

Our task is to enable the supplier to calculate the sum of maximum N measurements (i.e., $\sum_{i=1}^{N} X_t^i = \mathbf{X}_t$ in all t) coming from N different users while ensuring ε-differential privacy for each user. This is guaranteed if the supplier can only access $\mathbf{X}_t + \mathcal{L}(\lambda(t))$, where $\mathcal{L}(\lambda(t))$ [4] is the laplace noise calibrated to ε as it has been described in Section 5.1. There are (at least) 3 possible approaches to do this which are detailed as follows.

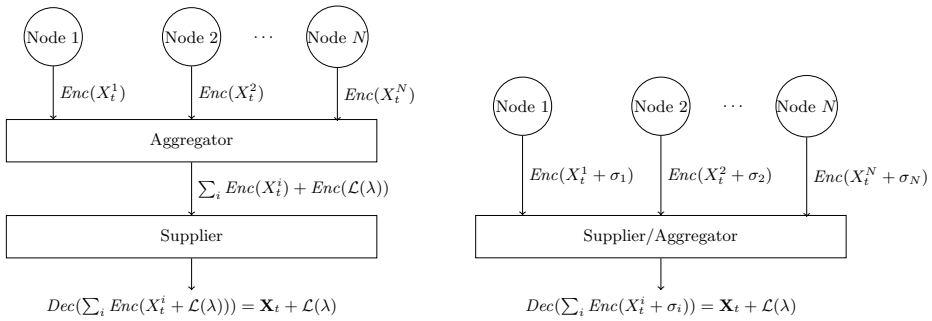
6.1 Fully decentralized approach (without aggregator)

Our first attempt is that each user adds some noise to its own measurement, where the noise is drawn from a Laplace distribution. In particular, each node i sends the value of $X_t^i + \mathcal{L}(\lambda)$ directly to the supplier in time t. It is easy to see that ε is guaranteed to all users, but in fact the final noise added to the aggregate (i.e., $\sum_{i=1}^{N} \mathcal{L}(\lambda)$) is N times larger than $\mathcal{L}(\lambda)$, and hence, the error is $\mu(t) = \frac{1}{\mathbf{X}_t + 1} \mathbb{E} |\sum_{i=1}^{N} \mathcal{L}(\lambda)| = \frac{N \cdot \lambda}{\mathbf{X}_t + 1}$. This means that $\mu(t)$ can be large.

6.2 Aggregation with a trusted aggregator

Our second attempt can be to aggregate the measurements of some users, and send the perturbed aggregate to the supplier. In particular, nodes are grouped into N sized clusters and each node of a cluster sends its measurement X_t^i to the (trusted) cluster aggregator, that

[4] We will use the notation λ instead of $\lambda(t)$ if the dependency on time is obvious in the context.



(a) Centralized approach: aggregation with trusted aggregator.

(b) Our approach: aggregation without trusted entity. If $\sigma_i = \mathcal{G}_1(N, \lambda) + \mathcal{G}_1(N, \lambda)$, where $\mathcal{G}_1$, $\mathcal{G}_2$ are i.i.d gamma noise, then $\sum_{i=1}^{N} \sigma_i = \mathcal{L}(\lambda)$.

Figure 1: The original and noisy measurements of user i, where the added noise is $\mathcal{G}_1(N, \lambda) - \mathcal{G}_2(N, \lambda)$ ($N = 100$, T_s is 10 min).

is a trusted entity different from the supplier. The aggregator computes $\mathbf{X}_t = \sum_{i=1}^{N} X_t^i$ and obtains $\hat{\mathbf{X}}_t = \mathbf{X}_t + \mathcal{L}(\lambda)$ by adding noise to the aggregate. This perturbed aggregate is then sent to the supplier as it is illustrated in Figure 1(a).

The utility of this approach is better than in the previous case, as the noise is only added to the sum and not to each measurement X_t^i. Formally, $\mu(t) = \frac{1}{\mathbf{X}_t+1}\mathbb{E}|\mathcal{L}(\lambda)| = \frac{\lambda}{\mathbf{X}_t+1}$. Similary, $\delta(t) = \frac{1}{\mathbf{X}_t+1} \cdot \sqrt{\mathbb{E}|\mathcal{L}(\lambda)|^2 - (\mathbb{E}|\mathcal{L}(\lambda)|)^2} = \frac{\lambda}{\mathbf{X}_t+1}$.

However, the main drawback of this approach is that the aggregator must be fully trusted since it receives each individual measurement from the users. This can make this scheme impractical in such cases where there is no such trusted entity.

6.3 Our approach: aggregation without trusted entity

Although the previous scheme is differential private, it only works if we trust the aggregator who adds a random value $\mathcal{L}(\lambda)$ to that aggregate before releasing it. If the aggregator omits adding the noise, our scheme will not be differential private.

Following the previous approach, we assume that nodes are grouped into clusters of N elements. Our scheme is composed of 2 steps: distributed generation of the laplacian noise and the encryption of individual measurements.

6.3.1 Distributed noise generation

Instead of adding noise at the aggregator, each user should add $\mathcal{L}(\lambda)$ to the aggregate in a distributed fashion as it is illustrated in Figure 1(b). The next lemma helps us to generate laplacian noise in a distributed way: it states that the Laplace distribution is divisible, as it can be constructed as the sum of i.i.d. gamma distributions. As this divisibility is infinite, it works for arbitrary number of users.

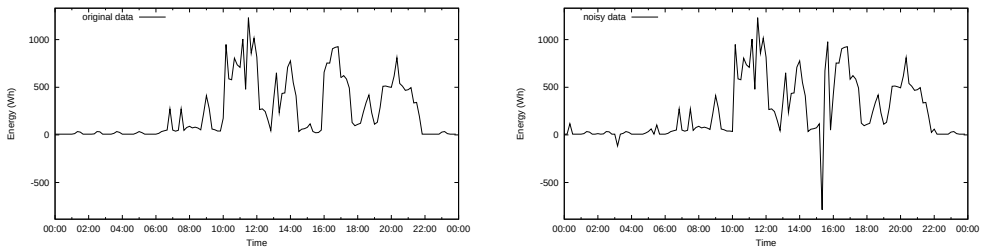
Lemma 1 (Divisibility of Laplace distribution [17]) *Let $\mathcal{L}(\lambda)$ denote a random variable which has a Laplace distribution with PDF $f(x, \lambda) = \frac{1}{2\lambda}e^{\frac{|x|}{\lambda}}$. Then the distribution of $\mathcal{L}(\lambda)$ is infinitely divisible. Furthermore, for every integer $n \geq 1$, $\mathcal{L}(\lambda) = \sum_{i=1}^{n}[\mathcal{G}_1(n, \lambda) - \mathcal{G}_2(n, \lambda)]$, where $\mathcal{G}_1(n, \lambda)$ and $\mathcal{G}_2(n, \lambda)$ are i.i.d. random variables having gamma distribution with PDF $g(x, n, \lambda) = \frac{(1/\lambda)^{1/n}}{\Gamma(1/n)}x^{\frac{1}{n}-1}e^{-x/\lambda}$ where $x \geq 0$.*

The lemma comes from the fact that $\mathcal{L}(\lambda)$ can be represented as the difference of two i.i.d exponential random variables with rate parameter $1/\lambda$. Moreover, $\sum_{i=1}^{n} \mathcal{G}_1(n,\lambda) - \sum_{i=1}^{n} \mathcal{G}_2(n,\lambda) = \mathcal{G}_1(1/\sum_{i=1}^{n} \frac{1}{n}, \lambda) - \mathcal{G}_2(1/\sum_{i=1}^{n} \frac{1}{n}, \lambda) = \mathcal{G}_1(1,\lambda) - \mathcal{G}_2(1,\lambda)$ due to the summation property of the gamma distribution[5]. Here, $\mathcal{G}_1(1,\lambda)$ and $\mathcal{G}_2(1,\lambda)$ are i.i.d exponential random variable with rate parameter $1/\lambda$ which completes the argument.

Our distributed sanitization algorithm is simple; user i calculates value $\hat{X}_t^i = X_t^i + \mathcal{G}_1(N,\lambda) - \mathcal{G}_2(N,\lambda)$ in slot t and sends it to the aggregator, where $\mathcal{G}_1(N,\lambda)$ and $\mathcal{G}_2(N,\lambda)$ denote two random values independently drawn from the same gamma distribution. Now, if the aggregator sums up all values received from the N users of a cluster, then $\sum_{i=1}^{N} \hat{X}_t^i = \sum_{i=1}^{N} X_t^i + \sum_{i=1}^{N} [\mathcal{G}_1(N,\lambda) - \mathcal{G}_2(N,\lambda)] = \mathbf{X}_t + \mathcal{L}(\lambda)$ based on Lemma 1.

6.3.2 Encryption

The previous step is not enough to guarantee privacy as only the sum of the measurements (i.e., $\hat{\mathbf{X}}_t$) is differential private but not the individual measurements. In particular, the aggregator has access to $\hat{X}_t^i$, and even if $\hat{X}_t^i$ is noisy, $\mathcal{G}_1(N,\lambda) - \mathcal{G}_2(N,\lambda)$ is usually insufficient to provide reasonable privacy for individual users if $N \gg 1$. This is illustrated in Figure 2, where an individual's noisy and original measurements slightly differ.



(a) X_t^i (b) $X_t^i + \mathcal{G}_1(N,\lambda) - \mathcal{G}_2(N,\lambda)$

Figure 2: The original and noisy measurements of user i, where the added noise is $\mathcal{G}_1(N,\lambda) - \mathcal{G}_2(N,\lambda)$ ($N = 100$, T_s is 10 min).

To address this problem, each contribution is encrypted using an additive homomorphic scheme such that the aggregator can only decrypt the sum of the individual values, and cannot access any of them.

It is easy to see that the utility of this scheme is identical to the utility in Section 6.1, and thus, it is optimal. In particular, $\mu(t) = \frac{1}{\mathbf{X}_t+1}\mathbb{E}|\mathbf{X}_t - \mathbf{X}_t + \sum_{i=1}^{n}[\mathcal{G}_1(N,\lambda) - \mathcal{G}_2(N,\lambda)]| = \frac{\mathbb{E}|\mathcal{L}(\lambda)|}{\mathbf{X}_t+1} = \frac{\lambda}{\mathbf{X}_t+1}$, and $\delta(t) = \frac{\lambda}{\mathbf{X}_t+1}$.

Finally, note that using our approach the aggregator and the supplier do need to be separate entities. The supplier can even play the role of the aggregator, as the encryption prevents it to access individual measurements, and the distributed generation of the noise ensures that it cannot manipulate the noise to remove the differential privacy guarantee of individuals.

[5]The sum of i.i.d. gamma random variables follows gamma distribution (i.e., $\sum_{i=1}^{n} \mathcal{G}(k_i,\lambda) = \mathcal{G}(1/\sum_{i=1}^{n} \frac{1}{k_i}, \lambda)$).

7 Protocol description

7.1 System setup

In our scheme, nodes are grouped into clusters of size N, where N is a parameter. The protocol requires the establishment of pairwise keys between each pair of the nodes inside a cluster that can be done by using traditional Diffie-Hellman key exchange [6] as follows:

1. When a node v_i is installed, it provides a self-signed DH component $g^{c_i} \pmod p$ and its certificate Cert_i to the supplier, where c_i is kept secret by v_i and g, p are public DH parameters.

2. Once all the nodes of a cluster are installed, or a new node is deployed, the supplier broadcasts the list of $(\mathsf{id}_i, g^{c_i} \pmod p, \mathsf{Cert}_i)$ $(1 \leq i \leq N)$, where id_i is the identity of node v_i.

3. Finally, each node v_i of the cluster can compute a pairwise key $K_{i,j}$ shared with any other node v_j by computing $g^{c_i \cdot c_j} \pmod p$. Note that no communication is required between v_i and v_j.

7.2 Smart meter processing

Each node v_i sends at time t its periodic measurement, X_t^i, to the supplier as follows:

Phase 1 (Data sanitization): Node v_i calculates value

$$\hat{X}_t^i = X_t^i + \mathcal{G}_1(N, \lambda) - \mathcal{G}_2(N, \lambda) \tag{1}$$

where $\mathcal{G}_1(N, \lambda)$ and $\mathcal{G}_2(N, \lambda)$ denote two random values independently drawn from the same gamma distribution and N is the cluster size.

Phase 2 (Data encryption): Each noisy data $\hat{X}_t^i$ is then encrypted using the CaTsMy homomorphic encryption scheme into $Enc(\hat{X}_t^i)$. The following extension is then applied: Each node, v_i, selects ℓ other nodes randomly, such that if v_i selects v_j, then v_j also selects v_i. Afterwards, both nodes generate a common dummy key k from their pairwise key $K_{i,j}$; v_i adds k to $Enc(\hat{X}_t^i)$ and v_j adds $-k$ to $Enc(\hat{X}_t^j)$. As a result, the aggregator cannot decrypt the individual ciphertexts (it does not know the dummy key k). However, it adds all the ciphertexts of a given cluster, the dummy keys cancel out and it retrieves the encrypted sum of the (noisy) contributions. The more formal description is as follows:

1. node v_i selects some nodes of the clusters randomly (we call them participating nodes) using a secure pseudo random function (PRF) such that if v_i selects v_j, then v_j also selects v_i. In particular, v_i selects v_j if mapping $PRF(K_{i,j}, r_1)$ to a value between 0 and 1 is less or equal than $\frac{w}{N-1}$, where r_1 is a public value changing in each slot. The PRF can be implemented using a stream cipher, such as RC4 (like in [4]). We denote by ℓ the number of selected participating nodes, and $\mathsf{ind}_i[j]$ (for $j = 1, \ldots, \ell$) denotes the index of the ℓ nodes selected by node v_i. Note that, for the supplier, the probability that v_i selects v_j is $\frac{w}{N-1}$ as it does not know $K_{i,j}$. The expected value of ℓ is w.

2. v_i computes for each of its ℓ participating nodes a *dummy key*. A dummy key between v_i and v_j is defined as $\mathsf{dkey}_{i,j} = (i-j)/|i-j| \cdot PRF(K_{i,j}, r_2) \pmod{\Delta}$, where $K_{i,j}$ is the key shared by v_i and v_j, and $r_2 \neq r_1$ is public value changing in each slot. Note that $\mathsf{dkey}_{i,j} = -\mathsf{dkey}_{j,i}$.
3. v_i then computes $Enc(\hat{X}_t^i) = \hat{X}_t^i + K_i' + \sum_{j=1}^{\ell} \mathsf{dkey}_{i,\mathsf{ind}_i[j]} \pmod{\Delta}$, where $K_i' \in [0, \Delta-1]$ is the keystream shared by v_i and the aggregator, and Δ is a large integer. Note that Δ must larger than the sum of all contributions (i.e., final aggregate) plus the laplacian noise.[6] In practice, if $p = \max_{t,i}(\hat{X}_t^i)$ (which can be easily estimated in advance) then Δ should be selected as $\Delta = 2^{\lceil log_2(p \cdot N) \rceil}$.

   Note that $\hat{X}_t^i$ is encrypted multiple times: it is first encrypted with the keystream K_i' and then with several dummy keys. K_i' is needed to prevent an eavesdropper to recover $\hat{\mathbf{X}}_t$, and can be established using the DH protocol as above. In particular, if the eavesdropper knows all $\hat{X}_t^i$ and they does not contain K_i', then summing them up the dummy keys cancel out and $\hat{\mathbf{X}}_t$ is obtained. The dummy keys are needed to prevent the aggregator (supplier) from retrieving $\hat{X}_t^i$.

$Enc(\hat{X}_t^i)$ is then sent to the aggregator (supplier).

7.3 Supplier processing

Phase 1 (Data aggregation): At each epoch, the supplier aggregates the N measurements received from the cluster smart meters by summing them, and obtains

$$Enc(\hat{\mathbf{X}}_t) = \sum_{i=1}^{N} Enc(X_t^i)$$

based on the homomorphic property of the encryption. In particular,

$$Enc(\hat{\mathbf{X}}_t) = \sum_{i=1}^{N} (\hat{X}_t^i + K_i') + \sum_{i=1}^{N} \sum_{j=1}^{\ell} \mathsf{dkey}_{i,\mathsf{ind}_i[j]} \pmod{\Delta}$$

where $\sum_{i=1}^{N} \sum_{j=1}^{\ell} \mathsf{dkey}_{i,\mathsf{ind}_i[j]} = 0$ because $\mathsf{dkey}_{i,j} = -\mathsf{dkey}_{j,i}$. Hence,

$$Enc(\hat{\mathbf{X}}_t) = \sum_{i=1}^{N} (\hat{X}_t^i + K_i') = \sum_{i=1}^{N} Enc(\hat{X}_t^i)$$

Phase 2 (Data decryption): The aggregator then decrypts the aggregated value by substracting the sum of the node's keystream, and retrieves the sum of the noisy measures:

$$Dec(\hat{\mathbf{X}}_t) = \sum_{i=1}^{N} Enc(\hat{X}_t^i) - \sum_{i=1}^{N} K_i' = \sum_{i=1}^{N} \hat{X}_t^i \pmod{\Delta}$$

where $\sum_{i=1}^{N} \hat{X}_t^i = \sum_{i=1}^{N} X_t^i + \sum_{i=1}^{N} \mathcal{G}_1(N, \lambda) - \sum_{i=1}^{N} \mathcal{G}_2(N, \lambda) = \sum_{i=1}^{N} X_t^i + \mathcal{L}(\lambda)$ based on Lemma 1.

[6] Note that the noise is a random value from an infinite domain and this sum might be larger than Δ. However, choosing sufficiently large Δ, the probability that the sum exceeds M can be made arbitrary small due to the exponential tail of the Laplace distribution.

The main idea of the scheme is that the aggregator is not able to decrypt the individual encrypted values because it does not know the dummy keys. However, by adding the different encrypted contributions, dummy keys cancel each other and the aggregator can retrieve the sum of the plaintext. The resulting plaintext is then the perturbed sums of the measurements, where the noise ensures the differential privacy of each user.

8 Adding robustness

We have assumed so far that all the N nodes of a cluster participated in the protocol. However, it might happen that, for several different reasons (e.g., node or communication failures) some nodes are not able to participate in each epoch. This would have two effects: first, security will be reduced since the sum of the noise added by each node will not be equivalent to $\mathcal{L}(\lambda)$. Hence, differential privacy may not be guaranteed. Second, the aggregator will not be able to decrypt the aggregated value since the sum of the dummy keys will not cancel out.

In this section, we extend our scheme to resist node failures. We propose a scheme which resists the failure of up to M out of N nodes, where M is a configuration parameter. We will study later the impact of the value M on the scheme performance.

8.1 Sanitization phase extension

In order to resist the failure of M nodes, each node should add the following noise to their individual measurement: $\mathcal{G}_1(N-M,\lambda) - \mathcal{G}_2(N-M,\lambda)$. Note that $\sum_{i=1}^{N-M}[\mathcal{G}_1(N-M,\lambda) - \mathcal{G}_2(N-M,\lambda)] = \mathcal{L}(\lambda)$. Therefore, this sanitization algorithm remains differential private, if at least $N-M$ nodes participate in the protocol. Note that in that case each node adds extra noise to the aggregate in order to ensure differential privacy even if less than M nodes fail to send their noise share to the aggregator.

8.2 Encryption phase extension

8.2.1 A naive approach

As described previously, all the dummy keys cancel out at the aggregator. However, this is not the case if not all the nodes participate in the protocol. In order to resist the failure of nodes, one can extend the encryption scheme with an additional round where the aggregator asks the participating nodes of non-responding nodes to send the missing dummy keys:

1. Once the aggregator received all contributions, it broadcasts the ids of the non-responding nodes. Note that the aggregator knows which nodes are inside cluster, and each node v_i should also attach id_i to its encrypted measurement which is sent to the aggregator.

2. Upon the reception of this message, each node v_i verifies whether any of the ids in the broadcast message are in its participating node list (i.e., it can be found in ind_i). For each of such id, the node sends the corresponding dummy key to the aggregator.

3. The aggregator then subtracts all received dummy key from $Enc(\hat{\mathbf{X}}_t)$ and retrieves $\sum_{i=1}^{N}(\hat{X}_t^i + K_i')$ which can be decrypted.

This approach has a severe problem: if the aggregator is untrusted, it can easily retrieve the measurement of a v_i: broadcasting its id in Step 2, the participating nodes of v_i reply with the dummy keys of v_i which can be removed from $Enc(\hat{X}_t^i)$. In particular, summing up the dummy keys coming from the participating nodes, the supplier obtains $\sum_{j=1}^{\ell} \mathsf{dkey}_{i,\mathsf{ind}_i[j]}$. Then, subtracting that and K_i' from the encrypted value of v_i: $Enc(\hat{X}_t^i) - \sum_{j=1}^{\ell} \mathsf{dkey}_{i,\mathsf{ind}_i[j]} - K_i' = \hat{X}_t^i$.

8.2.2 An advanced approach

In the naive approach, the aggregator needs the dummy keys themselves in order to remove them from the aggregate, but this also enables it to remove that from the node's encrypted value. In the following more advanced approach, each node adds a secret random value to its encrypted value before releasing it in the first round. This is needed to prevent the adversary to recover the noisy measurement through combining different messages of the nodes. Then, in the second round when the aggregator asks for the missing dummy keys, every node reveals its random keys along with the missing dummy keys that it knows:

1. Each node v_i sends

$$Enc(\hat{X}_t^i) = \hat{X}_t^i + K_i' + \sum_{j=1}^{\ell} \mathsf{dkey}_{i,\mathsf{ind}_i[j]} + C_i \pmod{\Delta}$$

   where C_i is the secret random key of v_i generated randomly in each round.

2. After receiving all measurements, the aggregator asks all nodes for their random keys and the missing dummy keys through broadcasting the id of the non-responding nodes.

3. Each node v_i verifies whether any ids in this broadcast message are in its participating node list, where the set of the corresponding participating nodes is denoted by S. Then, v_i replies with $\sum_{j \in S} \mathsf{dkey}_{i,\mathsf{ind}_i[j]} + C_i \pmod{\Delta}$.

4. The aggregator subtracts all received values from $Enc(\hat{\mathbf{X}}_t)$ which results in $\sum_{i=1}^{N}(\hat{X}_t^i + K_i')$, as the random keys as well as the dummy keys cancel out.

Note that as the supplier does not know the random keys, it cannot remove them from any messages but only from the final aggregate; adding each node's response to the aggregate all the dummy keys and secret random keys cancel out and the supplier obtains $\hat{\mathbf{X}}_t$. Although the supplier can recover v_i's measurement if it knows v_i's participating nodes (the supplier simply asks for all the dummy keys of v_i in the second round and subtracts v_i's response from its own encrypted value sent in the first round), we will show later that this probability can be made practically small by adjusting w and N correctly.

Note that the protocol fails if, for some reasons, a node does not send its random key to the aggregator (as only the node itself knows its random key, it cannot be reconstructed by other parties). However, it is very unlikely that a node between the two rounds fails, and an underlying reliable transport protocol helps to overcome communication errors.

Finally, also note that this random key approach always requires two rounds of communication (even if the aggregator receives all encrypted values correctly in the first round), as the random keys are needed to be removed from $Enc(\hat{\mathbf{X}}_t)$ in the second round.

8.3 Utility

If all N nodes participate in the protocol, the added noise will be larger than $\mathcal{L}(\lambda)$ which is needed to ensure differential privacy. In particular, $\sum_{i=1}^{N}[\mathcal{G}_1(N-M,\lambda)-\mathcal{G}_2(N-M,\lambda)] = \mathcal{L}(\lambda)+\sum_{i=1}^{M}[\mathcal{G}_1(N-M,\lambda)-\mathcal{G}_2(N-M,\lambda)]$, where the last summand is the extra noise needed to tolerate the failure of maximum M nodes. Clearly, this extra noise increases the error if all N nodes operate correctly and add their noise shares faithfully. In what follows, we calculate the error and its standard deviation if we add this extra noise to the aggregate.

Theorem 2 (Utility) *Let $\alpha = M/N$ and $\alpha < 1$. Then,*

$$\mu(t) \leq \frac{2}{B(1/2, \frac{1}{1-\alpha})} \cdot \frac{\lambda(t)}{\mathbf{X}_t + 1}$$

and

$$\sigma(t) \leq \sqrt{\left(\frac{2}{1-\alpha} - \frac{4}{B(1/2, \frac{1}{1-\alpha})^2}\right)} \cdot \frac{\lambda(t)}{\mathbf{X}_t + 1}$$

where $B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$ is the beta function.

The derivation can be found in Appendix A.
Based on Theorem 2,

$$\sigma(t) = \mu(t) \cdot \left(\frac{2}{B(1/2, \frac{1}{1-\alpha})}\right)^{-1} \cdot \sqrt{\left(\frac{2}{1-\alpha} - \frac{4}{B(1/2, \frac{1}{1-\alpha})^2}\right)}$$

Figure 3 shows the standard deviation of the error ($\sigma(t)$) in the function of the mean error ($\mu(t)$). Here, $\sigma(t)$ is always less or equal than $\mu(t)$. In particular, if $\alpha = 0$ (there are no malicious nodes and node failures), then $\sigma(t) = \mu(t)$. If $\alpha > 0$ then $\sigma(t) < \mu(t)$, and in general the greater α the greater the difference $\sigma(t) - \mu(t)$ is.

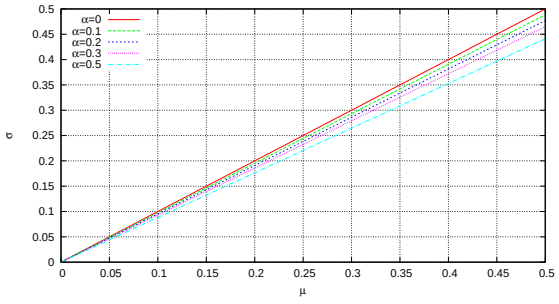


Figure 3: $\sigma(t)$ depending on $\mu(t)$ in case of different values of α

9 Security Analysis

9.1 Deploying malicious nodes

In the proposed scheme, each measurement is perturbed and encrypted. As a result, a honest-but-curious attacker cannot gain any information (up to $\varepsilon = \max_i X_t^i/\lambda(t)$) about

individual measurements in slot t. This is guaranteed by the encryption scheme and the added noise.

However, an active attacker, which deploys T malicious nodes, may be able to:

- reduce the noise level by limiting (or omitting) the gamma noise added by malicious nodes. As a result, the sum of the noise shares will not equal to the laplacian noise which can decrease the privacy of users. However, observe that due to the robustness property of our scheme detailed in Section 8 we add extra noise to tolerate M node failures. If we increase M with T, then we will add enough extra noise to tolerate this type of attack.

- decrypt $Enc(\hat{X}_t^i)$ of v_i and retrieve the perturbed data. As individual data is only weakly noised, the attacker might infer some information from them, and therefore, compromise privacy. However, as the (homomorphic) encryption scheme that we used is provably secure [], and nodes are assumed to be tamper-resistant, the only way to do that is to retrieve the dummy keys of v_i. Because the participating nodes are selected randomly for each message, this can only be achieved if **all** participating nodes of v_i are malicious **and** the supplier is also malicious (i.e., the adversary knows key K_i'). This happens if v_i does not select any honest participating node that has a probability of $(1 - \frac{w}{N-1})^{N-T-1}$. Therefore, setting w appropriately is crucial to defend against this type of attack. For instance, it is easy to check that if $N = 100$ and 50% of the nodes are malicious (which anyway should be a quite strong assumption), then setting w to 30 results in success probability of $1.8 \cdot 10^{-8}$. This means that if an epoch is 5 min long, then the adversary will compromise 1 measurement during 458 years in average.

  Finally, note also this is the success probability of the adversary in a single slot. As a result, a supplier that succeeds the previous attack only gets a single (noisy) measurement of the customer (corresponding to a single epoch). As a node selects different participating nodes in each slot, the probability that the adversary gets k different measurements of the node is $(1 - \frac{w}{N-1})^{k(N-T-1)}$, which is even smaller.

9.2 Lying supplier

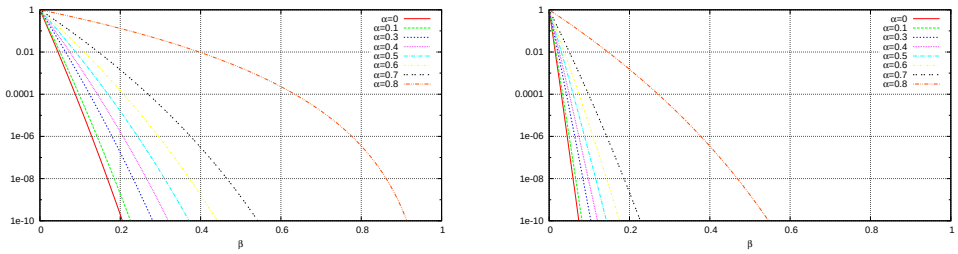
Lying about non-responding nodes

In addition to deploying malicious (fake) nodes, a malicious supplier can lie about the non-responding nodes. Indeed, a malicious supplier might pretend that a node v_i did not respond in the first round. As a result, and as described in Section 8.2.2, the participating nodes of v_i will disclose v_i's dummy keys that allows the supplier to decrypt v_i's contribution. However, v_i chooses its participating nodes randomly and changes them for each message, and hence, the supplier can only guess them.

In particular, the contribution of v_i is secured (i.e., cannot be decrypted), as long as there is at least one dummy key of v_i that is not known to the supplier. More precisely, in order to recover v_i's measurement, the supplier needs the sum of its random key C_i and its dummy keys. There are three types of v_i's dummy keys: the first is shared with a malicious node, and hence, known to the supplier. The second is asked from v_i by the supplier in the second round, and v_i replies with the sum of C_i and the asked keys. Finally, the rest is shared with honest participating nodes and not asked from v_i in the second round. Apparently, if v_i has at least one dummy key from the last group, its measurement cannot be recovered. In particular, if v_j

is a participating honest node of v_i and $\mathsf{dkey}_{i,\mathsf{ind}_i[j]}$ is not asked from v_i in the second round, it could be recovered only from v_j's messages. However, v_j sends $C_j + \mathsf{dkey}_{i,\mathsf{ind}_i[j]}$, where C_j is only known to v_j, and thus, the supplier cannot recover $\mathsf{dkey}_{i,\mathsf{ind}_i[j]}$.

Nevertheless, it may happen that v_i does not have any third-type dummy key (i.e., the supplier asks for all the dummy keys shared with honest nodes in the second round). Then, the supplier can easily recover v_i's measurement by combining its malicious dummy keys and the sum of the keys sent by v_i to the supplier. However, the supplier can only guess v_i's participating nodes[7]. Assuming that the supplier can ask v_i for maximum M dummy keys in the second round, the probability that all participating nodes of v_i are either malicious or specified as non-responding nodes by the supplier is less than $(1 - \frac{w}{N-1})^{N-(T+M)-1}$. Using $\alpha = (T + M)/N$ and $\beta = w/N$, then $(1 - \frac{w}{N-1})^{N-(T+M)-1} = (1 - \frac{\beta}{1-N^{-1}})^{N(1-\alpha)-1}$. This probability is depicted in Figure 4 depending on α, β and N.



(a) N=100 (b) N=300

Figure 4: Success probability of guessing participating nodes depending on β and different values of α and N.

Lying about cluster size

Another strategy for the supplier to compromise the privacy of users is to lie about the cluster size. If the supplier pretends that the cluster size N' is larger than it really is (i.e., $N' > N$), the noise added by each node will be underestimated. In fact, each node will calibrate its noise based on Formula (1) using N' instead of N. As a result, the aggregated noise at the supplier will be smaller than necessary to guarantee sufficient differential privacy.

In order to prevent this attack, a solution would be to set the cluster size to a fixed value. For example, all clusters should have a size of 100. Although simple and efficient, this solution is not flexible and might not be applicable to all scenarios. Another option is that the supplier publishes together with the list of cluster nodes, a self-signed certificate (containing a timestamp, the cluster id and the node information) of each node of the cluster. That way, each node could verify the cluster size and get information about its constituting nodes (recall that the DH certificate of an honest node described in Section 7.1 cannot be forged by the supplier).

[7]Note that *all* nodes send responses in the second round, and the randomness of C_i ensures that the supplier cannot gain any knowledge about the participating nodes of any nodes.

10 Simulation results

10.1 Available datasets

Although several research projects were performed in the last two decades aiming to characterize the electricity demand model of different households [10], they only provide high-level statistics: instead of the high resolution demand data, an average consumption value of the given hour of the day is published, where the average is taken on a few weeks long monitoring campaign. This data does not suit our purpose, as we are interested in the consumption values at finer sampling rates; we need the demand data of each household at every few minutes (like every 5, 10, 15 min). Intuitively, higher resolution aggregate has higher variance, and thus, it is much more privacy sensitive. In addition, if, for a given time slot, we take the average of several measurements taken in the given slot but on different days, then we also decrease the variance by "smoothing" out the real time consumption.

10.2 A high-resolution electricity demand model

Due to the lack of high-resolution real world data, we implemented a domestic electricity demand model [22] that can generate one-minute resolution synthetic consumption data of different households. This model follows a bottom-up approach and simulates each appliance in a household. The simulator includes 33 different appliances which are randomly assigned to every household based on real deployment statistics. The model also includes a separate lighting model which takes into account the level of natural daylight depending on the month of the year. The number of residents in each household is randomly selected between 1 and 5. A trace is associated to a household and generated as follows: (1) A number of active persons is selected according to some distribution derived from real statistics. This number may vary as some members can enter or leave the house. (2) A set of appliances is then selected and activated at different time of the day according to an other distribution, which was also derived from real statistics.

The input of the simulator is the number of households, the month of the day, and the type of the day (either a working or weekend day). The output is the power demand model (1-min profile) of all appliances in each household on the given day.

Using this simulator, we generated the 5, 10, and 15-min electricity consumption data of 3000 users.

10.3 Adding noise

Each trace was then sanitized according to our scheme. The noise added in each slot (i.e., $\lambda(t)$) is set to the maximum consumption in the slot (i.e., $\lambda(t) = \max_{1 \le i \le N} X_t^i$ where the maximum is taken on all users in the cluster). This amount of noise ensures $\varepsilon = 1$ indistinguishability for individual measurements in all slots. Although one can increase $\lambda(t)$ to get better privacy, the error will also increase. Note that the error $\mu_{\varepsilon'}(t)$ for other $\varepsilon' \neq \varepsilon$ values if $\mu_\varepsilon(t)$ is given is $\mu_{\varepsilon'}(t) = \frac{\varepsilon}{\varepsilon'} \cdot \mu_\varepsilon(t)$. We assume that $\lambda(t) = \max_i X_t^i$ is known a priori.

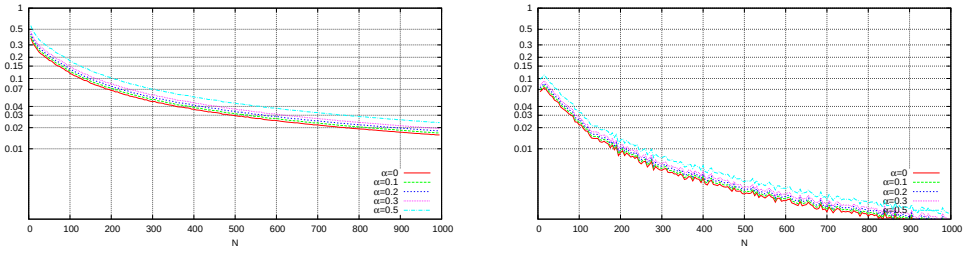
10.4 Error and the cluster size

We are interested in the dependence of the error ($\mu(t)$) on the cluster size (N). Recall that higher N results in higher $\mathbf{X}_t$ which decreases $\mu(t)$ in Definition 2.

10.4.1 Random clustering

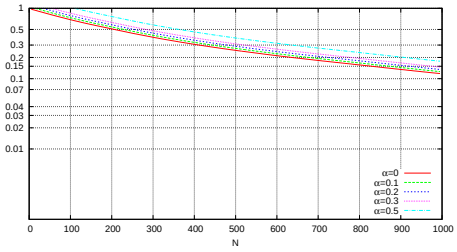
The most straightforward scheme to build N-sized clusters is to select N users uniformly at random. The advantage of this approach is that the users are not required to provide any information about their profile beyond $\hat{\mathbf{X}}_t$.

Figure 5(a) and 5(b) show the average error value and its standard deviation, resp., depending on the size of the cluster in case of different values of α. The average error of a given cluster size N is the average of $\mathsf{mean}_t(\mu(t))$ of all N-sized clusters[8]. Obviously, higher N causes smaller error, and thus, better utility. Assuming higher α, more extra noise is added, as it is described in Section 8.3, which also implies higher error. Interestingly, increasing the sampling period results in slight error decrease[9], hence, we only considered 10 min sampling period. Otherwise noted explicitly, we assume 10 min sampling period in the sequel. Finally, note that increasing the sampling rate decreases the accuracy of the monitoring, as each sent value is an aggregate. In this work, we are not concerned with this type of utility loss.



(a) Average error

(b) Standard deviation of the average error



(c) Maximum error

Figure 5: The error depending on N using random clustering. T_s is 10 min.

10.4.2 Consumption based clustering

As the error is $\lambda(t)/(\mathbf{X}_t+1)$, where $\lambda(t)$ is set to the maximum consumption in time t, one could get lower error if the maximum consumption is close to the mean of the measurements (i.e, $\mathbf{X}_t/N$) in every t inside a cluster. Hence, instead of randomly clustering users, a more clever approach is to cluster them based on the "similarity" of their consumption profiles. Intuitively, the measurements in similar profiles tend to be close, and thus, the difference

[8]In fact, the average error is approximated in Figure 5(a): we picked up 200 different clusters for each N, and plotted the average of their $\mathsf{mean}_t(\mu(t))$. 200 is chosen according to experimental analysis. Above 200, the average error do not change significantly.

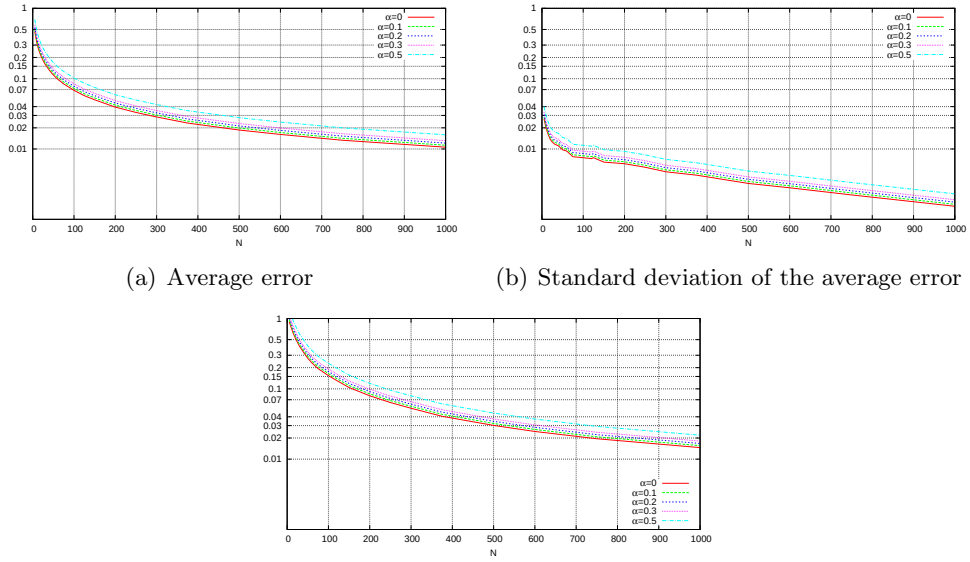
[9]This increase is less than 0.01 even if N is small when the sampling period is changed from 5 min to 15 min.

between the maximum consumption and $\mathbf{X}_t/N$ should also be smaller than in a random cluster.

Although there are multiple distance metrics to measure profile similarity, we just compare them based on the average daily consumption. In contrast to random clustering, this requires users to share their daily averages which may leak some information about their profile beyond $\hat{\mathbf{X}}_t$. However, the supplier does not need to know the high resolution profiles to calculate the daily averages; it can be derived from the (monthly) aggregate consumption of each user, which is though generally revealed for billing.

The N-sized clusters are created by calculating daily consumption levels such that the number of users whose daily average falls into the same level is exactly N. Then, all users form a cluster who are in the same level. This can be implemented by simply sorting the 3000 users according to their daily averages where each consecutive N users forms a cluster. If the size of the last sequence is less than N, we omit the corresponding users from further calculations.

Figure 6(a) and 6(b) show the average error and its deviation, resp., calculated identically to random clustering. Comparing Figure 6 and 5, consumption based clustering has lower error with 0.01-0.05 than the random one depending on N. For instance, while random clustering provides an average error of 0.13 with $N = 100$ users in a cluster, consumption based clustering has 0.07. The difference decreases as N increases. There are more significant differences between the standard deviations and the worst cases: on lower value of N, the standard deviation of the average error in random clustering is almost twice as large as in consumption based clustering (Figure 6(b) and 5(b)). To compute the worst case error, at a given N, the maximum error is computed in all slots, which is the highest cluster error that can occur in a slot with cluster size N. Then, the average of these maximum errors (the average is taken on all slots) are plotted in Figure 5(c) and 6(c). Apparently, the worst case error in random clustering is much higher than in consumption based clustering, as random clustering may put high and low consuming user into the same cluster.



(a) Average error

(b) Standard deviation of the average error

(c) Maximum error

Figure 6: The error depending on N using consumption based clustering. T_s is 10 min.

10.5 Boosting utility by lowering noise

Lowering noise has two main motivations. First, if there is a high difference between the maximum and the majority of the measurements inside a cluster, the error will be high. More specifically, if the distribution of the measurements tends to be heavy-tailed (i.e., there are outliers), then the noise set to the maximum measurement results in $\varepsilon = 1$ privacy for the outlier, but likely much lower indistinguishability for the other users in the cluster. Figure 7 shows what fraction of all users have lower indistinguishability than a specific ε. Indeed, in Figure 7, there is a large gap between the indistinguishability of the outlier and the majority of the users. Therefore, a reasonable decision can be to sacrifice the privacy of outliers to gain better utility, and add less noise. Although the *global* privacy bound increases and the indistinguishability of the outlier users can be large, the privacy bound of the majority are likely to remain still acceptable since their measurements are concentrated around a much lower value than the outlier measurements.

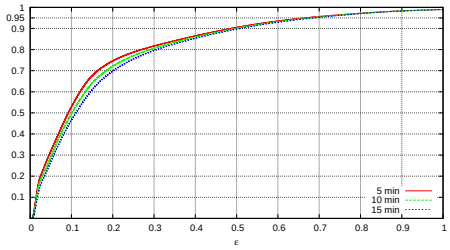
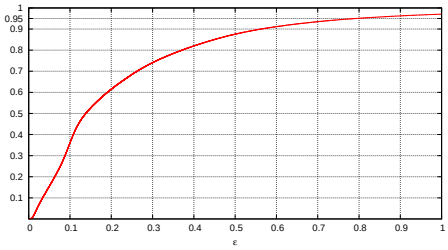


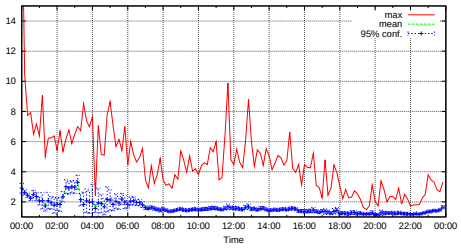
Figure 7: CDF of ε in case of different T_s. As the added noise is the maximum measurement inside a cluster, all users have lower indistinguishability than $\varepsilon = 1$.

The second motivation of lowering noise is to calibrate it to utility (instead of privacy). In practice, the supplier wishes to have a bound on the error with certain confidence which may allow to add much less noise than the maximum measurement. To bound the error $\mu(t)$ with certain confidence, the supplier needs to know the CDF of $\mathbf{X}_t = \sum_{i=1}^{N} X_t^i$. If N is large then $\mathbf{X}_t$ can be approximated by a gaussian random variable $\mathcal{N}(n \cdot \alpha_t, n \cdot \beta_t)$ based on the central limit theorem, where α_t and β_t are the mean and the variance of the distribution of measurements in time t, resp. If $\mu(t)$ must have a value of ρ in time t with probability p, then the supplier calculates the sum of measurements Ω above which integrating the distribution of the sums gives p. Formally, $\Phi^{-1}_{\alpha_t,\beta_t}(1-p) = \Omega$ from which $\lambda(t) = \rho \cdot (\Omega + 1)$, where $\Phi^{-1}_{\alpha_t,\beta_t}$ is the inverse CDF of $\mathcal{N}(\alpha_t, \beta_t)$.
▷ **TODO:** how α_t and β_t can be computed?

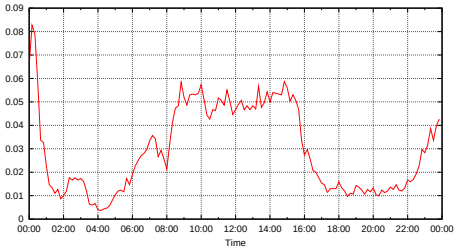
Figure 8 shows how the privacy changes by lowering the added noise to guarantee $\mu(t) = 0.05$ in all slots with probability $p = 0.95$. Comparing Figure 8(a) and Figure 7, the privacy bound does not change significantly for 90% of all users. Specifically, the added noise is still sufficient to guarantee $\approx \varepsilon = 0.7$ to them. The CDFs mainly differ when $\varepsilon > 1$. In particular, the mean value of ε in these cases (i.e., when a user has higher measurement than the added noise implying that $\varepsilon > 1$) varies between $\varepsilon = 3$ and $\varepsilon = 1$ (see Figure 8(b)). Finally, Figure 8(c) shows the fraction of users (out of 3000) who have $\varepsilon > 1$ in a given slot.



(a) CDF of ε when $\varepsilon < 1$.



(b) Statistics of ε when when $\varepsilon > 1$.



(c) Fraction of users who has $\varepsilon > 1$.

Figure 8: Lowering added noise to guarantee $\mu(t) = 0.05$ in all t with probability 0.95. T_s is 10 min.

10.6 Privacy over multiple slots

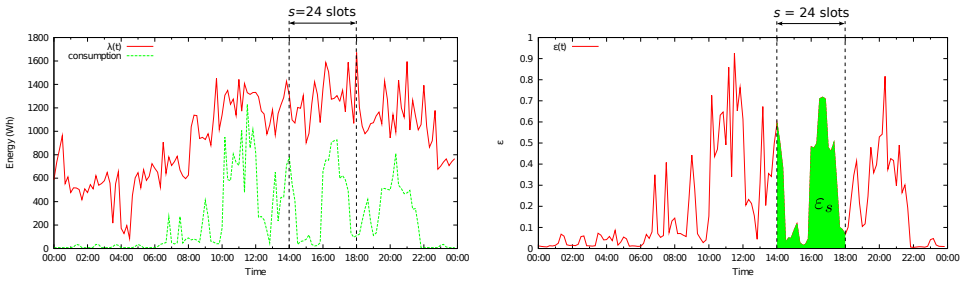
So far, we have considered the privacy of individual slots and added noise to guarantee $\varepsilon = 1$ privacy in each slot. However, the output are multiple aggregates as we release one aggregate per slot. For instance, if one watches TV along multiple slots, we have guaranteed that the adversary cannot tell if the TV is watched in any single slot (up to $\varepsilon = 1$), but it may be able to tell that the TV is watched in a given time period having s slots ($s > 1$) (the privacy bound of this is $\varepsilon_s = \varepsilon \cdot s$ due to the composition property of differential privacy). Based on Theorem 1, we need to add noise $\lambda(t) = \sum_{i=1}^{s} \max_i X_t^i$ to *each* aggregate to guarantee $\varepsilon_s = 1$ bound in consecutive s slots, which results in much higher error than in the case of $s = 1$ that we have assumed so far.

Obviously, using the LPA technique, we cannot guarantee reasonably low error if s increases, as the necessary noise $\lambda(t) = \sum_{i=1}^{s} \max_i X_t^i$ can be large. In order to keep the error $\lambda(t)/\sum_{i=1}^{N} X_t^i$ low while ensuring better privacy than $s \cdot \varepsilon$, one can increase the number of users inside each cluster (i.e., N). In the sequel, we investigate what privacy we can guarantee to some activities by adding the noise described in Section 10.3.

To illustrate the calculation of the bound ε_s with a given noise $\lambda(t)$ when $s > 1$, consider Figure 9. The consumption of a user i (i.e., X_t^i) and the added noise $\lambda(t)$ is plotted in Figure 9(a). Suppose we want to know what privacy user i has between 14:00 and 18:00. The bound $\varepsilon(t) = X_t^i/\lambda(t)$ in a single slot t is plotted in Figure 9(b). The bound ε_s for the $s = 24$ slots between 14:00 (84th slot) and 18:00 (108th slot) is $\sum_{t=84}^{108} \varepsilon(t)$ based on the composition property of differential privacy, which is 7.52 in the current example. In general, $\varepsilon_s(t) = \sum_{i=t}^{t+s} \varepsilon(i)$.

Figure 10 shows what privacy a user has in average in our dataset depending on the cluster size in case of different values of s. For a given N and s, we computed $\max_t \varepsilon_s(t)$ for all users. The average of all these maximum bounds are plotted.

As the cluster size increases, the bound becomes smaller. This is because the cluster



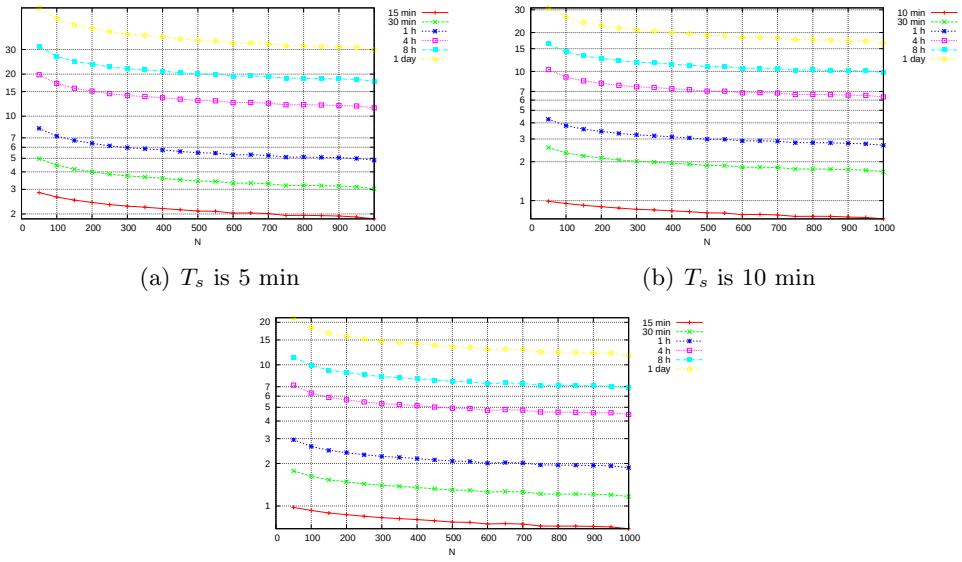
(a) Consumption and the added noise $\lambda(t)$

(b) $\varepsilon(t)$ is the bound in slot t. The bound ε_s between 14:00 and 18:00 is the shaded region below the curve of $\varepsilon(t)$.

Figure 9: Privacy in a given time window.

will have more users, and the maximum consumption (i.e., the noise) inside a cluster also increases (recall that increasing N in consumption based clustering increases the average consumption level of each cluster). Also note that increasing the sampling period also increases privacy.

▷ **TODO:** why?



(a) T_s is 5 min

(b) T_s is 10 min

(c) T_s is 15 min

Figure 10: Privacy of all appliances in s long time windows (where s is 10 min, 15 min, 30 min, 1 h, 4 h, 8 h, 1 day) using different sampling periods.

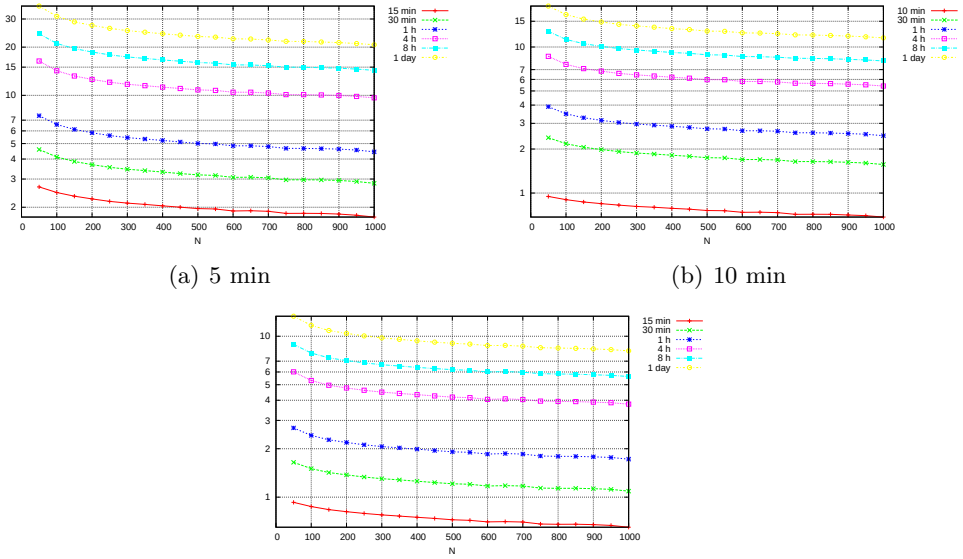
10.6.1 Privacy of appliances

We interpret the privacy of different activities (such as watching TV or cooking) between time t_1 and t_2 analogously to the example illustrated before. The only difference is that the consumption represents the sum consumption of the appliances (instead of the consumption of all appliances in Figure 9(a)) which are used to perform the activity. For instance, if

we are interested in what privacy the user had when he watched a game in the TV last night between 18:00 and 20:00 by adding noise $\lambda(t)$, we need to calculate $\sum_{t=108}^{120} \varepsilon(t)$, where $\varepsilon(t) = \{\text{TV's consumption in } t\}/\lambda(t)$.

Table 1 in Appendix B summarizes the average bound (computed identically as in Figure 10) of the main appliances used in our simulator. In addition, the table contains the standard deviation of $\max_t \varepsilon_s(t)$ and its maximum value in the whole dataset. The appliances are divided into two major groups: the usage of active appliances indicate that the user is at home and uses the appliance (their consumption significantly changes during their active usage such as iron, vacuum, kettle, etc.), whereas passive appliances have more or less identical consumption regardless the user is at home (like fridge, freezers, storage heater, etc.)[10]. In general, appliances having lower consumption threats privacy less than devices with higher energy demands. Obviously, ε_s increases when s increases since an appliance is used more frequently within longer periods.

Finally, consider the case when we want to hide the usage of any active appliances within any s long period. Following from the description detailed above, this is equivalent to answer the question if the user was at home in any s long period. The average bounds are depicted in Figure 6. Comparing Figure 11 and 10, there is no considerable differences between them. Consequently, there is no signficant change in terms of privacy if we consider all or only active appliances, as a profile is primarily shaped by active appliances (because they typically consume much more than passive appliances).



(a) 5 min

(b) 10 min

(c) 15 min

Figure 11: Privacy of active appliances in s long time windows using different sampling periods.

[10]We admit that this grouping is subjective. E.g., users may switch off freezers when they leave for holiday. However, in that case, s has typically much larger values than one day.

11 Conclusion and Future work

References

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

[2] R. Anderson and S. Fuloria. Who controls the off switch? In *Proceedings of the IEEE SmartGridComm*, June 2010.

[3] J.-M. Bohli, C. Sorge, and O. Ugus. A privacy model for smart metering. In *Proceedings of the IEEE International Conference on Communications (ICC 2010)*, 2010.

[4] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *ACM/IEEE Mobiquitous Conference, San Diego, USA*, July 2005.

[5] R. Cramer, I. Damgard, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, 2001.

[6] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[7] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of EUROCRYPT*, 2006.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd IACR Theory of Cryptography Conference*, 2006.

[9] C. Efthymiou and G. Kalogridis. Smart grid privacy via anonymization of smart metering data. In *Proceedings of IEEE SmartGridComm*, October 2010.

[10] EIE/05/124/SI2.419657. Residential monitoring to decrease energy use and carbon emissions in europe (remodece). In *http://remodece.isr.uc.pt/*, 2008.

[11] P. A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Proceedings of the 4th International Conference on Financial Cryptography (FC'00)*, pages 90–104, 2001.

[12] F. D. Garcia and B. Jacobs. Privacy-friendly energy-metering via homomorphic encryption. In *Proceedings of the 6th Workshop on Security and Trust Management (STM 2010)*, 2010.

[13] O. Goldreich. `http:/www.wisdom.weizmann.ac.il/oded/PS/prot.ps`.

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

[15] G. Kalogridis, C. Efthymiou, S. Denic, T. A. Lewis, and R. Cepeda. Privacy for smart meters: Towards undetectable appliance load signatures. In *Proceedings of the First IEEE International Conference on Smart Grid Communications (SmartGridComm 2010)*, October 2010.

[16] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *Proceedings of WWW 2009*, 2009.

[17] S. Kotz, T. J. Kozubowski, and K. Podgorski. *The Laplace distribution and generalizations: a revisit with applications to Communications, Economics, Engineering, and Finance*. Birkhauser, 2001.

[18] H. Lam, G. Fung, and W. K. Lee. A novel method to construct taxonomy electrical appliances based on load signatures. *IEEE Transactions on Consumer Electronics*, 53(2):653–660, December 2007.

[19] 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 2010)*, 2010.

[20] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the ACM SIGMOD/PODS Conference (SIGMOD 2010)*, June 2010.

[21] A. Rial and G. Danezis. Privacy-preserving smart metering. In *Technical Report, MSR-TR-2010-150*. Microsoft Research, 2010.

[22] I. Richardson, M. Thomson, D. Infield, and C. Clifford. Domestic electricity use: A high-resolution energy demand model. *Energy and Buildings*, 42:1878–1887, 2010.

[23] E. Shi, T. Chan, E. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *Network and Distributed System Security Symposium (NDSS). To appear.*, 2011.

A Proof of Theorem 2 (Utility)

Lemma 2 (Integral property of the Bessel function [17]) *Let*

$$K_\vartheta(x) = \frac{1}{2}\left(\frac{x}{2}\right)^\vartheta \int_0^\infty t^{-\vartheta-1}\exp\left(-t-\frac{x^2}{4t}\right)dt, \qquad x>0$$

define the modified Bessel function of the third kind with index $\vartheta \in \mathbb{R}$. For any $\gamma > 0$ and γ, ν such that $\gamma + 1 \pm \nu > 0$

$$\int_0^\infty x^\gamma K_\nu(ax)dx = \frac{2^{\gamma-1}}{a^{\gamma+1}}\Gamma\left(\frac{1+\gamma+\nu}{2}\right)\Gamma\left(\frac{1+\gamma-\nu}{2}\right)$$

Lemma 3 *Let $\mathcal{G}_1, \mathcal{G}_2$ be i.i.d gamma random variables with parameters (n, λ). Then,*

$$\mathbb{E}|\mathcal{G}_1(n,\lambda) - \mathcal{G}_2(n,\lambda)| = \frac{2\lambda}{B\left(\frac{1}{2},\frac{1}{n}\right)} \tag{2}$$

and

$$Var|\mathcal{G}_1(n,\lambda) - \mathcal{G}_2(n,\lambda)| = \left(\frac{2}{n} - \frac{4}{B\left(\frac{1}{2},\frac{1}{n}\right)^2}\right)\lambda^2 \tag{3}$$

where $B(x,y)$ is the beta function defined as $B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$.

Proof (of Lemma 3) Consider $\mathcal{Y} = \mathcal{G}_1 - \mathcal{G}_2$. The characteristic function of $\mathcal{Y}$ is

$$\phi_{\mathcal{Y}}(t) = \left(\frac{1}{1+i\lambda t}\right)^{\frac{1}{n}} \cdot \left(\frac{1}{1-i\lambda t}\right)^{\frac{1}{n}} = \left(\frac{1}{1+\lambda^2 t^2}\right)^{\frac{1}{n}}$$

which is a special case of the characteristic function of the Generalized Asymetric Laplace distribution (GAL) with parameters $(\theta, \kappa, \omega, \tau)$:

$$\phi_{GAL}(t) = e^{i\theta t}\left(\frac{1}{1+i\frac{\sqrt{2}}{2}\omega\kappa t}\right)^\tau \cdot \left(\frac{1}{1-i\frac{\sqrt{2}}{2\kappa}\omega t}\right)^\tau$$

where $\theta = 0, \kappa = 1, \omega = \sqrt{2}\lambda$, and $\tau = 1/n$. The density function of $GAL(\theta, \kappa, \omega, \tau)$ when $\theta = 0$ and $\kappa = 1$ is

$$f_{GAL}(x) = \frac{\sqrt{2}}{\omega^{\tau+1/2}\Gamma(\tau)\sqrt{\pi}}\left(\frac{|x|}{\sqrt{2}}\right)^{\tau-1/2} K_{\tau-1/2}(\sqrt{2}|x|/\omega)$$

where $K_{\tau-1/2}(\frac{\sqrt{2}}{\omega}|x|)$ is the Bessel function defined in Lemma 2. In addition,

$$\mathbb{E}|\mathcal{Y}| = \int_{-\infty}^\infty |x| f_{GAL}(x)dx = 2\cdot\int_0^\infty x\frac{\sqrt{2}}{\omega^{\tau+1/2}\Gamma(\tau)\sqrt{\pi}}\left(\frac{x}{\sqrt{2}}\right)^{\tau-1/2} K_{\tau-1/2}(\sqrt{2}x/\omega)dx$$

which follows from the symmetry property of $f_{GAL}(x)$ ($\phi_{\mathcal{Y}}(t)$ is is real valued). After reformulation, we have

$$\mathbb{E}|\mathcal{Y}| = \frac{2\sqrt{2}}{\sqrt{2}^{\tau-1/2}\omega^{\tau+1/2}\Gamma(\tau)\sqrt{\pi}}\int_0^\infty x^{\tau+1/2}K_{\tau-1/2}(\sqrt{2}x/\omega)dx$$

Now, we can apply Lemma 2 for the integral and we obtain

$$\mathbb{E}|\mathcal{Y}| = \sqrt{2} \cdot w \cdot \frac{\Gamma(\tau + \frac{1}{2})}{\Gamma(\frac{1}{2})\sqrt{\pi}}$$

after simple derivation. Using that $\sqrt{\pi} = \Gamma(1/2)$ and $B(x, y) = \frac{\Gamma(x)(y)}{\Gamma(x+y)}$, we have

$$\mathbb{E}|\mathcal{Y}| = \frac{\sqrt{2}}{B(1/2, \tau)} \cdot w$$

Applying $\omega = \sqrt{2}\lambda$ and $\tau = 1/n$, we arrive at Equation (2).

To prove Equation (3), consider that

$$Var(|\mathcal{Y}|) = \mathbb{E}|\mathcal{Y}|^2 - [\mathbb{E}|\mathcal{Y}|]^2$$

where

$$\mathbb{E}|\mathcal{Y}|^2 = \mathbb{E}(\mathcal{Y}^2) = \mathbb{E}(\mathcal{G}_1^2) + \mathbb{E}(\mathcal{G}_2^2) - 2 \cdot \mathbb{E}(\mathcal{G}_1) \cdot \mathbb{E}(\mathcal{G}_2)$$

Using that $\mathbb{E}(\mathcal{G}_1^2) = \mathbb{E}(\mathcal{G}_2^2) = (1/n^2 + 1/n)\lambda^2$, we obtain Equation (3).

Now, we can easily prove Theorem 2.

Proof (of Theorem 2)

$$\mathbb{E}|\sum_{i=1}^{N}(X_t^i - \hat{X}_t^i)| =$$

$$= \mathbb{E}|\sum_{i=1}^{N}\mathcal{G}_1(N - M, \lambda) - \sum_{i=1}^{N}\mathcal{G}_2(N - M, \lambda)| =$$

(using that $\sum_{i=1}^{n}\mathcal{G}(k_i, \lambda) = \mathcal{G}(1/\sum_{i=1}^{n}\frac{1}{k_i}, \lambda)$)

$$= \mathbb{E}|\mathcal{G}_1(1 - M/N, \lambda) - \mathcal{G}_2(1 - M/N, \lambda)| =$$

(using $\alpha = M/N$ and applying Lemma 3)

$$= \frac{2}{B(1/2, \frac{1}{1-\alpha})}\lambda$$

The standard deviation $\sqrt{Var|\sum_{i=1}^{N}(X_t^i - \hat{X}_t^i)|}$ can be derived identically.

B Privacy of some ordinary appliances

	Appliance	$s = 30\,\text{min}$			$s = 1\,\text{h}$			$s = 4\,\text{h}$			$s = 8\,\text{h}$			$s = 24\,\text{h}$		
		mean	*dev*	*max*	*mean*	*dev*	*max*	*mean*	*dev*	*max*	*mean*	*dev*	*max*	*mean*	*dev*	*max*
Active appliances	Lighting	0.91	1.28	17.87	1.29	1.37	18.84	2.68	1.82	19.38	3.63	2.29	21.49	4.89	2.97	25.37
	Cassette / CD Player	0.02	0.04	0.79	0.04	0.04	0.81	0.05	0.05	0.82	0.07	0.05	0.88	0.09	0.07	0.96
	Hi-Fi	0.10	0.17	4.43	0.16	0.19	4.59	0.17	0.20	4.62	0.18	0.21	4.62	0.19	0.21	4.62
	Iron	0.75	1.81	42.91	0.82	1.82	42.99	0.92	1.83	42.99	1.00	1.86	42.99	1.02	1.89	42.99
	Vacuum	1.67	7.59	134.54	1.70	7.59	134.54	1.82	7.58	134.54	1.90	7.60	134.54	1.94	7.63	134.54
	Fax	0.04	0.10	1.55	0.04	0.10	1.55	0.04	0.10	1.55	0.05	0.10	1.56	0.05	0.10	1.56
	Personal computer	0.21	0.32	7.48	0.34	0.36	7.48	0.83	0.49	7.48	1.09	0.58	7.53	1.42	0.83	8.37
	Printer	0.07	0.30	7.78	0.08	0.31	7.78	0.09	0.31	7.78	0.10	0.31	7.78	0.11	0.31	7.83
	TV	0.15	0.47	7.41	0.22	0.48	7.45	0.37	0.52	7.45	0.45	0.58	8.37	0.50	0.63	8.37
	VCR / DVD	0.05	0.16	2.81	0.07	0.17	2.84	0.10	0.17	2.89	0.13	0.18	2.95	0.14	0.19	3.01
	TV Receiver box	0.03	0.11	2.12	0.05	0.11	2.21	0.08	0.12	2.32	0.10	0.13	2.40	0.11	0.14	2.42
	Hob	1.90	9.58	132.86	1.96	9.58	132.86	2.15	9.57	132.86	2.28	9.59	132.86	2.34	9.67	132.86
	Oven	1.50	3.91	96.19	1.58	3.92	96.19	1.74	3.94	96.19	1.85	3.97	96.19	1.91	4.07	98.51
	Microwave	1.13	4.23	82.73	1.20	4.24	82.73	1.26	4.24	82.73	1.29	4.27	83.17	1.31	4.29	83.57
	Kettle	0.55	2.71	63.59	0.59	2.71	63.59	0.72	2.73	63.87	0.83	2.76	64.22	1.02	2.79	64.22
	Small cooking (group)	0.25	1.61	26.16	0.25	1.61	26.16	0.26	1.61	26.16	0.27	1.61	26.16	0.27	1.62	26.16
	Dish washer	0.93	2.67	55.64	1.49	2.67	55.64	1.78	2.71	55.64	1.97	2.95	60.15	2.03	2.97	60.15
	Tumble dryer	2.57	8.05	152.33	3.93	8.16	154.99	5.24	8.20	155.08	6.30	8.33	155.08	7.01	8.68	155.08
	Washing machine	1.23	1.43	31.57	1.30	1.45	31.72	1.96	1.63	33.24	2.55	1.76	33.24	3.07	2.07	34.62
	Washer dryer	1.82	1.08	19.22	3.17	1.33	19.27	4.70	1.99	25.82	6.39	2.38	25.82	7.92	3.49	33.66
	E-INST	1.47	1.12	6.54	1.93	1.15	6.54	3.47	1.16	7.58	4.70	1.49	9.00	7.06	2.13	10.99
	Electric shower	2.13	14.78	249.24	2.16	14.78	249.24	2.28	14.78	249.24	2.34	14.78	249.24	2.38	14.80	249.24
Passive app.	DESWH	3.34	14.01	249.29	4.04	14.04	251.01	6.13	14.06	253.21	7.83	14.23	255.20	10.85	14.57	257.76
	Storage heaters	3.22	0.32	3.96	5.64	0.56	6.95	20.20	1.99	24.87	30.45	4.23	41.48	30.45	4.23	41.48
	Elec. space heating	1.64	0.85	6.14	2.86	1.07	7.54	7.49	2.15	13.03	8.50	2.49	14.57	10.06	4.08	26.25
	Chest freezer	0.61	0.74	15.94	0.61	0.74	15.95	1.39	0.92	17.20	1.85	1.07	18.10	2.55	1.24	18.96
	Fridge freezer	0.91	0.39	7.56	0.91	0.40	7.61	2.19	0.95	8.67	2.94	1.25	10.58	4.07	1.61	11.69
	Refrigerator	0.44	0.22	3.83	0.45	0.23	4.00	1.06	0.49	4.77	1.40	0.64	5.68	1.92	0.80	6.50
	Upright freezer	0.67	0.39	8.37	0.67	0.39	8.42	1.63	0.80	9.09	2.16	1.03	10.99	2.98	1.31	11.98

Table 1: ε_s of different appliances in case of different s. $N = 100$ and the sampling period is 10 min.