I have a DREAM!
(DiffeRentially privatE smArt Metering)

Gergely Ács and Claude Castelluccia

INRIA Rhone Alpes, Montbonnot, France
{gergely.acs, claude.castelluccia}@inrialpes.fr

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.

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 [2]. 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) [11] 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 [14]. 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 [15]. 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 trusted third party (i.e., an aggregator). We were able to avoid the use of a trusted third 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., 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 can retrieve information about any user consumption only up to a predefined threshold, no matter what 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 an α fraction of all nodes of a cluster collude with the supplier, where α is a security parameter.
– We implemented a new electricity trace generation tool based on [19] which generates realistic, one-minute resolution synthetic consumption data of different households. We used this simulator to evaluate the performance and privacy of our proposal.

Because of space constraint, the security analysis of our scheme is not included in this paper. This analysis is however included in the longer version of this paper [1]. This extended version also includes additional performance results.

2 Related Work

Several papers addressed the privacy problems of smart metering in the recent past [8, 15, 2, 16, 3, 4, 18, 10]. However, only a few of them have proposed technical solutions to protect users' privacy. In [2, 3], the authors discuss the different security aspects of smart metering and the conflicting interests among stakeholders. The privacy of billing is considered in [18, 15]. Seemingly, the privacy of monitoring the sum consumption of multiple users may be solved by simply anonymizing individual measurements like in [8] 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 [12]. Moreover, these techniques require an existing trusted third party who performs anonymization. The authors in [4] 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 sum up at the aggregation, the final noise makes the aggregate useless. In contrast to this, [10] uses homomorphic encryption to guarantee privacy for individual measurements. However, the aggregate is not perturbed which means that it is not differential private.

Three closely related works to ours are [17, 20, 6]. [6] describes protocols for generating shares of random noise which is secure against malicious participants. However, it requires communication between users and it uses expensive secret sharing techniques resulting in high overhead in case of large number of users. In [17], 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 [9] for homomorphic encryption which is much more expensive compared to [5] 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 only need one extra round. In addition, our solution does not rely on expensive public key cryptography during aggregation.

A recent paper [20] proposes another technique to privately aggregate time series data. This work differs from ours as follows: (1) they use a Diffie-Hellman-based encryption scheme, whereas our construction is based on a more efficient construction that only use modular additions. This approach is better adapted to resource constrained devices like smart meters. (2) Although [20] does not require the establishment (and storage) of pairwise keys between nodes as opposed to our approach, it is unclear how [20] can be extended to tolerate node and communication failures. By contrast, our scheme is more robust, as the encryption key of non-responding nodes is known to other nodes in the network that can help to recover the aggregate. (3) Finally, [20] uses a different noise generation method from ours, but this technique 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. Although δ can be arbitrarily small, this also decreases the utility. By contrast, in our scheme, $\delta = 0$ while ensuring nearly optimal utility.

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 measures the electricity consumption of the user in every T_p long period, and, using the communication network, sends the measurement to the aggregator at the end of every slot (in practice, T_p is around 1-30 minutes). Note that the communication and distribution network can be the same (e.g., when PLC technology is used to transfer data). 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_p; 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., $\sum_{i=1}^{N} X_t^i \overset{\mathsf{def}}{=} \mathbf{X}_t$).

As in [4], 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 [3]. 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 [3]. As in [3], 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. Messages exchanged between the supplier and the meters are authenticated using pairwise MACs [1]. Smart meters are assumed to have bidirectional communication channel (using some wireless or PLC technology) with the aggregator, but the meters cannot communicate with each other. We suppose that nodes may (randomly) fail, and in these cases, cannot send their measurements to the aggregator. However, nodes are supposed to use some reliable transport protocol to overcome the transient communication 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 [18, 15].

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 include the level of power consumption, periodicity, and duration.

In this paper we consider a *dishonest-but-non-intrusive (DN) adversary*. A DN adversary may not follow the protocol correctly and is allowed to provide false information to manipulate the collected data. He may also collude with some (malicious) smart meters. However, he is not allowed to access or modify the distribution network to mount

[1] Please refer to [16] for a more detailed discussion about key management issues in smart metering systems.

attacks. In particular, he is not allowed to install wiretapping devices to eavesdrop on the victim's consumption.

3.3 Privacy model

We use differential privacy [7] 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.

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 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), like 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.

3.4 Output perturbation: achieving differential privacy

Let's say that we want to publish in a differentially private way the output of a function f. The following theorem says that this goal can be achieved by perturbing the output of f; simply adding a random noise to the value of f, 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 the sum of all its inputs, the sensitivity is the maximum value that an input can take.

Theorem 1 ([7]). *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*[2].

[2] 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 two consecutive slots. The measurements of U_1, U_2 and U_3 are $(X_1^1 = 300, X_2^1 = 300)$, $(X_1^2 = 100, X_2^2 = 400)$, and $(X_1^3 = 50, X_2^3 = 150)$, resp. The nodes want differential privacy for the released sums with at least a $\varepsilon = 0.5$. Based on Theorem 1, they need to add $\mathcal{L}(\lambda = \max_i \sum_t X_t^i/0.5 = 1200)$ noise to the released sum in **each** slot. This noise ensures $\varepsilon = \sum_t X_t^1/\lambda = 0.5$ individual indistinguishability for U_1, $\varepsilon = 0.42$ for U_2, and $\varepsilon = 0.17$ for U_3. Hence, the global $\varepsilon = 0.5$ bound is guaranteed to all. Another interpretation is that U_1 has $\varepsilon_1 = X_1^1/\lambda = 0.25$, $\varepsilon_2 = X_2^1/\lambda = 0.25$ privacy in each individual slot, and $\varepsilon = \varepsilon_1 + \varepsilon_2 = 0.5$ considering all two slots following from the composition property of differential privacy.

3.5 Utility definition

Let $f : \mathbb{D} \to \mathbb{R}$. 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 (Laplacian Perturbation Algorithm). 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.*

In the rest of this paper, the terms ”utility” and ”error” are used interchangeably.

4 Secure aggregation without aggregator: an overview

Our scheme enables 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 smart meters 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 $\mathcal{L}(\lambda(t))$ [3] is the Laplacian noise calibrated to ε as it has been described in Section 3.4.

A simple solution would be to rely on an aggregator that aggregates the N samples and adds Laplacian noise before forwarding the result to the supplier. Although this scheme would be differential private, it only works if the aggregator is trusted. In particular, the scheme will not be secure if the aggregator omits to add the noise.

Our scheme, instead, does not rely on any centralized aggregator. The noise is added by each smart meter on their individual data and encrypted in such a way that the aggregator can only compute the (noisy) aggregate. Note that with our approach the aggregator and the supplier do need to be separate entities. The supplier can even play the role

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

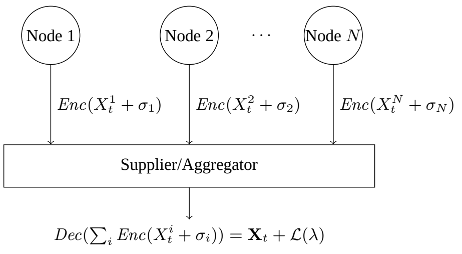


Fig. 1. Our approach: aggregation without trusted entity. If $\sigma_i = \mathcal{G}_1(N,\lambda) + \mathcal{G}_2(N,\lambda)$, where $\mathcal{G}_1$, $\mathcal{G}_2$ are i.i.d gamma noise, then $\sum_{i=1}^N \sigma_i = \mathcal{L}(\lambda)$.

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.

Our proposal is composed of 2 main steps: distributed generation of the Laplacian noise and encryption of individual measurements. These 2 steps are described in the remainder of this section.

4.1 Distributed noise generation: a new approach

In our proposal, the Laplacian noise is generated in a fully distributed way as is illustrated in Figure 4. We use the following lemma that states that the Laplace distribution is divisible and 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 [13]). *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[4]. 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.

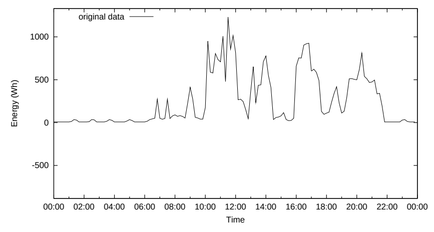
Our distributed sanitization algorithm is simple; user i calculates value $\hat{X}_t^i = X_t^i + \mathcal{G}_1(N,\lambda) - \mathcal{G}_2(N,\lambda)$ in slot t and sends it to the aggregator, where $\mathcal{G}_1(N,\lambda)$ and $\mathcal{G}_2(N,\lambda)$ denote two random values independently drawn 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.

The utility of our distributed scheme is defined as $\mu(t) = \frac{1}{\mathbf{X}_t+1}\mathbb{E}|\mathbf{X}_t - \mathbf{X}_t + \sum_{i=1}^n [\mathcal{G}_1(N,\lambda) - \mathcal{G}_2(N,\lambda)]| = \frac{\mathbb{E}|\mathcal{L}(\lambda)|}{\mathbf{X}_t+1} = \frac{\lambda}{\mathbf{X}_t+1}$, and $\delta(t) = \frac{\lambda}{\mathbf{X}_t+1}$.

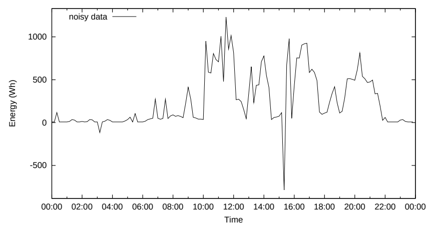
[4] The sum of i.i.d. gamma random variables follows gamma distribution (i.e., $\sum_{i=1}^n \mathcal{G}(k_i,\lambda) = \mathcal{G}(1/\sum_{i=1}^n \frac{1}{k_i}, \lambda)$).

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

Fig. 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_p is 10 min).

To address this problem, each contribution is encrypted using a modulo addition-based encryption scheme, inspired by [5], such that the aggregator can only decrypt the sum of the individual values, and cannot access any of them. In particular, let k_i denote a random key generated by user i inside a cluster such that $\sum_{i=1}^{N} k_i = 0$, and k_i is not known to the aggregator. Furthermore, $Enc()$ denotes a probabilistic encryption scheme such that $Enc(p,k,m) = p + k \mod m$, where p is the plaintext, k is the encryption key, and m is a large integer. The adversary cannot decrypt any $Enc(\hat{X}_t^i, k_i, m)$, since it does not know k_i, but it can easily retrieve the noisy sum by adding the encrypted noisy measurements of all users; $\sum_{i=1}^{N} Enc(\hat{X}_t^i, k_i, m) = \sum_{i=1}^{N} \hat{X}_t^i + \sum_{i=1}^{N} k_i = \sum_{i=1}^{N} \hat{X}_t^i \mod m$. If $z = \max_{i,t}(\hat{X}_t^i)$ then m should be selected as $m = 2^{\lceil \log_2(z \cdot N) \rceil}$ [5]. The generation of k_i is described in Section 5.2.

5 Protocol description

5.1 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 nodes inside a cluster that can be done by using traditional Diffie-Hellman key exchange as follows. When a node v_i is installed, it provides a self-signed DH component and its certificate to the supplier. Once all the nodes of a cluster are installed, or a new node is deployed, the supplier broadcasts the certificates and public DH components of all nodes. Finally, each node v_i of the cluster can compute a pairwise key $K_{i,j}$ shared with any other node v_j in the networks.

5.2 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 $\hat{X}_t^i = X_t^i + \mathcal{G}_1(N,\lambda) - \mathcal{G}_2(N,\lambda)$, 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 into $Enc(\hat{X}_t^i)$ using the modulo addition-based encryption scheme detailed in Section 4.2. The following extension is then applied to generate the encryption keys: 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 cluster 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. 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 $\mathsf{dkey}_{i,j} = (i-j)/|i-j| \cdot PRF(K_{i,j}, r_2)$, 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 $Enc(\hat{X}_t^i) = \hat{X}_t^i + K_i' + \sum_{j=1}^{\ell} \mathsf{dkey}_{i,\mathsf{ind}_i[j]} \pmod m$, where K_i' is the keystream shared by v_i and the aggregator which can be established using the DH protocol as above, and m is a large integer (see [5]). Note that m must be larger than the sum of all contributions (i.e., final aggregate) plus the Laplacian noise.[5]
   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 ensure confidentiality between a user and the aggregator. The dummy keys are needed to prevent the aggregator (supplier) from retrieving $\hat{X}_t^i$.
4. $Enc(\hat{X}_t^i)$ is sent to the aggregator (supplier).

5.3 Supplier processing

Phase 1 (Data aggregation): At each slot, the supplier aggregates the N measurements received from the cluster smart meters by summing them, and obtains $\sum_{i=1}^{N} Enc(X_t^i)$.

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

In particular, $Enc(\hat{\mathbf{X}}_t) = \sum_{i=1}^{N}(\hat{X}_t^i + K_i') + \sum_{i=1}^{N}\sum_{j=1}^{\ell} \mathsf{dkey}_{i,\mathsf{ind}_i[j]} \pmod m$, 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 subtracting the sum of the node's keystream, and retrieves the sum of the noisy measures: $\sum_{i=1}^{N} Enc(\hat{X}_t^i) - \sum_{i=1}^{N} K_i' = \sum_{i=1}^{N} \hat{X}_t^i \pmod m$ 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.

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.

Complexity: Let b denote the size of the pairwise keys (i.e., $K_{i,j}$). Our scheme has $O(N \cdot b)$ storage complexity, as each node needs to store $\ell \leq N$ pairwise keys. The computational overhead is dominated by the encryption and the key generation complexity. The encryption is composed of $\ell \leq N$ modular addition of $\log_2 m$ bits long integers, while the key generation needs the same number of PRF executions. This results in a complexity of $O(N \cdot (\log_2 m + c(b)))$, where $c(b)$ is the complexity of the applied PRF function. [6]

6 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.

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

[6] For instance, if $\log_2 m = 32$ bits (which should be sufficient in our application), $b = 128$, and $N = 1000$, a node needs to store 16 Kb of key data and perform maximum 1000 additions along with 1000 subtractions (for modular reduction) on 32 bits long integers, and maximum 1000 PRF executions. This overhead should be negligible even on constrained embedded devices.

the protocol. Note that in that case each node adds extra noise to the aggregate in order to ensure differential privacy even if fewer than M nodes fail to send their noise share to the aggregator.

Encryption phase extension The encryption phase consists of two rounds. In the first round, each node adds a secret random value to its encrypted value before releasing it. In the second round, every node reveals its random value along with the missing dummy keys that it knows:

1. Each node v_i sends $Enc(\hat{X}_t^i) = \hat{X}_t^i + K_i' + \sum_{j=1}^{\ell} \mathsf{dkey}_{i,\mathsf{ind}_i[j]} + C_i \pmod m$ 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 $\sum_{j \in S} \mathsf{dkey}_{i,\mathsf{ind}_i[j]} + C_i \pmod m$.
4. The aggregator subtracts all received values from $\sum_{i=1}^{N} Enc(\hat{X}_t^i)$ 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.

The main idea of this scheme is that C_i prevents the supplier to recover $\hat{X}_t^i$ by combining the messages of nodes. Indeed, if v_i did not add C_i to its messages in Step 1 and 3, the supplier could easily get $\hat{X}_t^i$ by subtracting the responses of v_i's participating nodes (and K_i' that it knows), received in Step 3, from $Enc(\hat{X}_t^i)$, which is received in Step 1. However, since the supplier does not know the random keys, it cannot remove them from any messages but only from the final aggregate; subtracting the response of each node, received in Step 3, from the aggregate, all the dummy keys and secret random keys cancel out and the supplier obtains $\hat{\mathbf{X}}_t$. Although the supplier can still recover $\hat{X}_t^i$ if it knows v_i's participating nodes (the supplier simply asks for all the dummy keys of v_i in Step 2 and subtracts v_i's response in Step 4 from $Enc(\hat{X}_t^i)$), this probability can be made practically small by adjusting w and N correctly (see [1] for details).

Utility evaluation 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. *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 the full version of this paper [1]. Based on Theorem 2, $\sigma(t) = \mu(t) \cdot \left(\frac{2}{B(1/2, \frac{1}{1-\alpha})}\right)^{-1} \cdot \sqrt{\left(\frac{2}{1-\alpha} - \frac{4}{B(1/2, \frac{1}{1-\alpha})^2}\right)}$. It is easy to check that $\sigma(t)$ 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 $\sigma(t) < \mu(t)$ but $\sigma(t) \approx \mu(t)$.

7 Simulation results

7.1 Electricity trace simulator

Due to the lack of high-resolution real world data, we implemented an electricity trace simulator that can generate realistic one-minute resolution synthetic consumption traces. It is an extended version of the simulator developed in [19]. The simulator includes 33 different appliances. 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.

Using this simulator, we generated 3000 electricity traces corresponding to different households, where the number of residents in each household was randomly selected between 1 and 5. Each trace was then sanitized according to our scheme. The noise added in each slot (i.e., $\lambda(t)$) was 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.

7.2 Performance analysis: error according to the cluster size

The error introduced by our scheme depends on the cluster size N. In this section, we present how the error varies according to N. Table 1 shows 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[7]. Obviously, higher N causes smaller error. Furthermore, a high α results in larger noise added by each meters, as described in Section 6, which also implies larger error. Interestingly, increasing the sampling period (i.e., T_p) results in slight error decrease[8], hence, we only considered 10 min sampling period. Otherwise noted explicitly, we assume 10 min sampling period in the sequel.

[7] In fact, the average error is approximated in Table 1: 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 does not change significantly.

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

N	$\alpha = 0$		$\alpha = 0.1$		$\alpha = 0.3$		$\alpha = 0.5$	
	mean	*dev*	*mean*	*dev*	*mean*	*dev*	*mean*	*dev*
100	0.118	0.021	0.135	0.023	0.150	0.026	0.177	0.032
300	0.047	0.004	0.050	0.005	0.054	0.006	0.070	0.007
500	0.029	0.002	0.031	0.002	0.036	0.002	0.044	0.003
800	0.019	0.001	0.020	0.001	0.023	0.001	0.028	0.001
1000	0.015	0.0008	0.016	0.0008	0.019	0.001	0.023	0.001

Table 1. The error depending on N and α using random clustering. The sampling period is 10 min.

7.3 Privacy evaluation

Privacy over multiple slots So far, we have considered the privacy of individual slots, i.e. added noise to guarantee $\varepsilon = 1$ privacy in each slot of size 10 minutes. However, a trace is composed of several slots. For instance, if a user watches TV during multiple slots, we have guaranteed that an adversary cannot tell if the TV is watched in any particular slot (up to $\varepsilon = 1$). However, by analysing s consecutive slots corresponding to a given period, it may be able to tell whether the TV was watched during that period (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, of course, results in 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 $\varepsilon_s = s \cdot \varepsilon$, one can increase the number of users inside each cluster (i.e., N).

Let's say that we want to compute the privacy of a user i between 14:00 and 18:00. If $\varepsilon(t) = X_t^i/\lambda(t)$ denotes the bound in a single slot t, then, based on the composition property of differential privacy, the bound ε_s for the $s = 24$ slots between 14:00 (84th slot) and 18:00 (108th slot) is $\sum_{t=84}^{108} \varepsilon(t)$. In general, $\varepsilon_s(t) = \sum_{i=t}^{t+s} \varepsilon(i)$.

Table 2 shows what average privacy of a user, in our dataset, as a function of the cluster size and value s. As the cluster size increases, the privacy bound decreases (i.e. privacy increases). The reason is that when the cluster size increases, the maximum consumption also increases with high probability. Since the noise is calibrated according to the maximum consumption within the cluster, it will be larger. This results in better privacy.

N	$s = 3$ (30 min)		$s = 24$ (4 h)		$s = 48$ (8 h)		$s = 144$ (24 h)	
	mean	*dev*	*mean*	*dev*	*mean*	*dev*	*mean*	*dev*
100	2.34	0.40	9.05	2.59	14.18	3.94	26.24	4.52
300	2.02	0.44	7.60	2.69	11.81	4.14	20.95	4.62
500	1.87	0.45	7.04	2.76	10.90	4.25	19.01	4.85
800	1.76	0.45	6.64	2.79	10.27	4.34	17.56	5.10
1000	1.67	0.47	6.35	2.87	9.83	4.47	16.55	5.40

Table 2. ε_s of users considering all appliances depending on N and s. T_p is 10 min.

Privacy of appliances In the previous section, we analysed how a user's privacy varies over time. In this section, we consider the privacy of the different appliances. For example, we aim at answering the following question: *what was my privacy when I was watching TV last evening between 18:00 and 20:00?* In order to compute the corresponding privacy (i.e. ε_s), we compute $\sum_{t=108}^{120} \varepsilon(t)$, where $\varepsilon(t) = \{\text{TV's consumption in } t\}/\lambda(t)$.

We summarized some of the appliance privacy [9] in Table 3. Each value is computed by averaging the privacy provided in our 3000 traces.

The appliances can be 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 (like fridge, freezers, storage heater, etc.) have more or less identical consumption regardless the user is at home or not.

	$s = 3$ (30 min)		$s = 24$ (4 h)		$s = 48$ (8 h)		$s = 144$ (24 h)	
	mean	*dev*	*mean*	*dev*	*mean*	*dev*	*mean*	*dev*
Lighting	0.91	1.28	2.68	1.82	3.63	2.29	4.89	2.97
Cassette / CD Player	0.02	0.04	0.05	0.05	0.07	0.05	0.09	0.07
Vacuum	1.67	7.59	1.82	7.58	1.90	7.60	1.94	7.63
Personal computer	0.21	0.32	0.83	0.49	1.09	0.58	1.42	0.83
TV	0.15	0.47	0.37	0.52	0.45	0.58	0.50	0.63
Microwave	1.13	4.23	1.26	4.24	1.29	4.27	1.31	4.29
Kettle	0.55	2.71	0.72	2.73	0.83	2.76	1.02	2.79
Washing machine	1.23	1.43	1.96	1.63	2.55	1.76	3.07	2.07
DESWH	3.34	14.01	6.13	14.06	7.83	14.23	10.85	14.57
Storage heaters	3.22	0.32	20.20	1.99	30.45	4.23	30.45	4.23
Refrigerator	0.44	0.22	1.06	0.49	1.40	0.64	1.92	0.80

Table 3. ε_s of different appliances in case of different s. $N = 100$ and T_p is 10 min. The name of active devices are in bold.

Previous tables show two different, and conflicting, results. Table 2 shows that it may actually be difficult to hide the presence of activities in a household. In fact, computed ε values are quite high, even for large clusters. However, results presented in Table 3 are more encouraging. They show that, although, it might be difficult to hide a user's presence, it is still possible to hide his actual activity. In fact, appliances privacy bounds (ε values) are quite small, which indicates that an adversary will have difficulty telling whether the user is, for example, using his computer or watching TV during a given period of time. Furthermore, results show that it is even more difficult for an adversary to tell when a given activity actually started. Finally, we recall that in order to keep the error $\lambda(t)/\sum_{i=1}^{N} X_t^i$ low while ensuring better privacy one can always increase the number of users inside each cluster. For instance, doubling N from 100 to 200 allows to double the noise while keeping approximately the same error value (0.118 in Table 1 if

[9] Because of space constraint, we are only able to display a small sample of our results. A larger table can be found in [1]

$\alpha = 0$). This results in much better privacy, since, on average, doubling the noise halves the privacy parameter ε_s.

Although more work and research is needed, we believe this is a encouraging result for privacy. Protecting users' privacy against smart metering systems might not be a dream after all!

Acknowledgements

The work presented in this paper was supported in part by the European Commission within the STREP WSAN4CIP project. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsement of the WSAN4CIP project or the European Commission.

References

1. G. Acs and C. Castelluccia. I have a DREAM! (DIffeRentially PrivatE smart Metering). In *Technical Report*, `http://planete.inrialpes.fr/~ccastel/PAPERS/IH_TR.pdf`, 2011.
2. R. Anderson and S. Fuloria. On the security economics of electricity metering. In *Proceedings of the WEIS*, June 2010.
3. R. Anderson and S. Fuloria. Who controls the off switch? In *Proceedings of the IEEE SmartGridComm*, June 2010.
4. J.-M. Bohli, C. Sorge, and O. Ugus. A Privacy Model for Smart Metering. In *Proceedings of IEEE ICC*, 2010.
5. C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient Aggregation of Encrypted Data in Wireless Sensor Networks. In *ACM/IEEE Mobiquitous Conference*, 2005.
6. C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our Data, Ourselves: Privacy via Distributed Noise Generation. In *Proceedings of EUROCRYPT*, 2006.
7. C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the 3rd IACR TCC*, 2006.
8. C. Efthymiou and G. Kalogridis. Smart Grid Privacy via Anonymization of Smart Metering Data. In *Proceedings of IEEE SmartGridComm*, October 2010.
9. P. A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Proceedings of FC*, pages 90–104, 2001.
10. F. D. Garcia and B. Jacobs. Privacy-friendly Energy-metering via Homomorphic Encryption. In *Proceedings of the STM*, 2010.
11. G. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, December 1992.
12. A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing Search Queries and Clicks Privately. In *Proceedings of WWW 2009*, 2009.
13. S. Kotz, T. J. Kozubowski, and K. Podgorski. *The Laplace distribution and generalizations*. Birkhauser, 2001.
14. 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.
15. A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *Proceedings of ACM Buildsys*, 2010.

16. R. Anderson and S. Fuloria and F. Alvarez and K. McGrath. Key Management for Substations: Symmetric Keys, Public Keys or No Keys? In *IEEE PSCE*, 2011.
17. V. Rastogi and S. Nath. Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption. In *Proceedings of the ACM SIGMOD*, June 2010.
18. A. Rial and G. Danezis. Privacy-Preserving Smart Metering. In *Technical Report, MSR-TR-2010-150*. Microsoft Research, 2010.
19. 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.
20. E. Shi, T. Chan, E. Rieffel, R. Chow, and D. Song. Privacy-Preserving Aggregation of Time-Series Data. In *Proceedings of NDSS*, February 2011.