BES* - Differentially Private and Distributed Event Aggregation in Advanced Metering Infrastructures

Vincenzo Gulisano, Valentin Tudor, Magnus Almgren, Marina Papatriantafilou
{vincenzo.gulisano, tudor, magnus.almgren, ptrianta}@chalmers.se
Department of Computer Science and Engineering,
Chalmers University of Technology, Göteborg, Sweden

ABSTRACT

Significant challenges for online event aggregation in the context of Cyber-Physical Systems stem from the computational requirements of their distributed nature, as well as from their privacy concerns. In the context of the latter, *differential privacy* has gained popularity because of its strong privacy protection guarantees, holding against very powerful adversaries. Despite such strong guarantees, though, its adoption in real-world applications is limited by the privacy-preserving noise it introduces to the analysis, which might compromise its usefulness.

We investigate the above problem from a system-perspective in the context of Advanced Metering Infrastructures, providing strong privacy guarantees together with useful results for event aggregation taking into account the distributed nature of such systems. We present a streaming-based framework, *Bes*, and propose methods to limit the noise introduced by differential privacy in real-world scenarios, thus reducing the resulting utility degradation, while still holding against the adversary model adhering with the original definition of differential privacy.

We provide a thorough evaluation based on a fully implemented *Bes* prototype and conducted with real energy consumption data. We show how a large number of events can be aggregated in a private fashion with low processing latency by a single-board device, similar in performance to the devices deployed in Advanced Metering Infrastructures.

Keywords

Differential privacy, data streaming, advanced metering infrastructures

1. INTRODUCTION

Cyber-Physical Systems serve critical aspects of our modern lives and it is crucial to secure these systems while preserving the privacy of the users' data. In this paper we focus on such an important system, which monitors and controls the modern electrical grid. Through the presence of devices that measure and communicate in a *continuous* fashion, cyber-physical systems such as advanced metering infrastructures (AMIs) allow energy providers to share fine-grained energy consumption statistics with citizens (e.g., to increase their awareness) and with third-parties providing a broad range of applications such as intrusion and fraud detection or enhanced energy management ones (e.g., energy consumption forecast or demand-response applications).

The emergence of differential privacy. In AMIs, the concept of privacy takes on a new meaning when fine-grained energy consumption statistics are shared with the public. If a private person wants to feed the aggregated consumption of his appliances to a third-party tool monitoring his consumption trends, can he be sure that said tool will not run statistical methods such as Non-intrusive appliance Load Monitoring [16] and infer which (possibly sensitive) appliances he owns? Even worse, when sharing the aggregated consumption statistics of a group of households, can AMIs prevent a burglar from inferring, based on such statistics, the time when energy consumption is minimal (i.e., the time when specific households most likely will be empty)?

The peculiarity of these privacy leaks is their resistance to privacy mechanisms such as encryption (which could prevent leaks while consumption readings are reported e.g. from users to energy providers) or homomorphic encryption (which could be leveraged to perform data aggregation directly on the ciphertext by untrusted third-party applications), which do not hold once statistics computed over sensitive data are intentionally shared with the public. In the prevention of privacy leaks in such a context, *differential privacy* has gained compelling interest because of its protection guarantees against strong adversaries. In a nutshell, *differential privacy* sacrifices the accuracy of a statistic by adding an appropriate level of noise to it, thus preventing the adversary from gaining knowledge about the individual values contributing to the statistic. Although it offers strong guarantees, the adoption of differential privacy in real-world applications is limited by the introduced noise, which might compromise its utility.

Challenges. Far-reaching complementary challenges must be addressed for advanced metering infrastructures to share statistics with the public or with sensitive applications such as fraud detection or energy consumption forecast:

i) *preserving of the privacy* of the individuals behind such statistics,

ii) *maximization of the statistics' utility* given the accuracy degradation introduced by differential privacy, and

iii) *efficient and distributed computation of such statistics*, leveraging the heterogeneous embedded devices composing AMIs, coping with the large data volumes they generate continuously and providing statistics in a real-time fashion.

*Named after the Egyptian deity protector of households.

CPSS'16, May 30-June 03 2016, Xi'an, China

DOI: http://dx.doi.org/10.1145/2899015.2899021

Based on these observations, the following question drove our research: *can we release statistics from AMIs, while (i) providing differential privacy, (ii) minimizing the degradation it introduces and (iii) computing such statistics in an efficient and distributed fashion?*

Contributions. We can give an affirmative answer to the above question. We present *Bes*[1], a framework for differentially private streaming aggregation (summation) of energy consumption measurements with an error that is both small and tunable. We make the following contributions:

i) We provide a method that, based on the trade-offs between the utility maximization and the privacy preservation of data, can maximize the utility of a differentially private statistic by controlling its aggregation parameters. From a broader perspective, this allows for differential privacy to be practically leveraged in existing cyber-physical systems such as AMIs.

ii) We provide a streaming-based design and implementation of *Bes* based on Apache Storm [26], a state-of-the-art stream processing engine which allows for fast distributed online analysis.

iii) We provide a thorough evaluation, based on a real prototype and conducted with events collected from a real-world advanced metering infrastructure. As shown, *Bes* allows for accurate privacy-preserving aggregation (with errors that can be lower than 10%) of thousands of events per second on inexpensive single-board devices representative of the ones that can form cyber-physical systems such as AMIs.

Our novel approach is a decomposition of the problem in two subproblems: (i) calibrating the privacy-preserving noise by computing approximated sums in a differentially private streaming fashion and (ii) making the approximation process itself differentially private. This decomposition admits efficient methods for solving each of the two subproblems and combines into a low-error, differentially private computation.

To the best of our knowledge, *Bes* is the first framework demonstrating that differential privacy can indeed be leveraged by AMIs' applications demanding for accurate aggregation (with errors that can be lower than 10%) of thousands of events per second run by inexpensive single-board devices (representative of devices that are already present in the advanced metering infrastructure or devices that can be deployed with negligible costs). While focusing on energy consumption events, *Bes* motivates the adoption of differentially private distributed event aggregation also in other sensitive data-intensive applications (e.g., in vehicular monitoring applications, where sensitive information is constituted by users' locations and traveling habits, or in manufacturing and marketing applications, where sensitive information is constituted by users' product demands and interests).

The rest of the paper is organized as follows. We introduce the system model, the problem statement and differential privacy in Section 2. We present *Bes*' differentially private aggregation mechanism in Sections 3 and 4. We overview the implementation of *Bes* in Section 5 and discuss its evaluation in Section 6. Section 7 discusses the related work while Section 8 concludes the paper.

2. SYSTEM AND PROBLEM DESCRIPTION

This section discusses the AMI system model, the problem tackled by *Bes*, and gives an overview of differential privacy notions employed by *Bes*.

[1]Available at https://github.com/vincenzo-gulisano/Bes

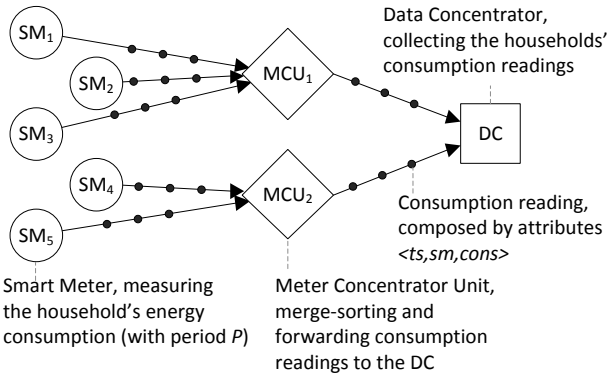


Figure 1: System model, composed by Smart Meters, Meter Concentrator Units and a centralized Data Collector.

2.1 System Model

As shown in Figure 1, our system consists of three entities: *Smart Meters* (SMs), *Meter Concentrator Units* and a centralized *Data Collector*. In AMIs, SMs installed at each household are used by energy providers (i.e., the Data Collector) to gather *energy consumption readings* via Meter Concentrator Units. A stream of energy consumption readings is defined as an unbounded sequence of *events*, sharing a schema $\langle ts, sm, cons \rangle$. Attribute ts represents the time when an event is generated and forwarded by the smart meter sm. Attribute $cons$ represents the consumption measured over a given period P. As discussed in [13], without loss of generality we assume events belonging to the same stream are delivered by each smart meter in timestamp order. When multiple streams are fed to a Meter Concentrator Unit (or to the Data Collector), we also assume they are merge-sorted into a single stream of timestamp-sorted events in order to be aggregated in a deterministic fashion [5]. In the following, we use the terms *event* and *energy consumption reading* interchangeably.

When sharing statistics about aggregated energy consumption (e.g., with customers or third-party applications), the aggregation is usually computed over *time-based sliding windows* covering portions of the most recent events [4] due to the higher relevance of fresh over historical events. The window size *WS* and window advance *WA* parameters specify the extent of a window and the amount of information discarded each time the latter slides. The group-by parameter *GB* can be used to group results sharing one or more attributes (e.g., to produce statistics for each separate SM sm rather than statistics aggregating events from distinct SMs). Table 1 summarizes the aggregation parameters.

Sample event aggregation. Figure 2 presents a sample stream of energy consumption readings reported by two SMs with a periodicity *P* of one hour, the evolution of the time-based sliding window (given a window size *WS* of two hours and a window advance *WA* of one hour) and the resulting stream of output events, grouped by each SM. Input events carry a timestamp, the id of the SM and its energy consumption in kWh. Output events carry a timestamp, the id of the SM and the aggregated consumption in kWh for the last 2 hours. As it can be noted, the results for a given window of time (e.g., the results for the window spanning the time period *[00:00-02:00)*) are produced as soon as the first event that no longer contributes to such window is received (e.g., when the event with timestamp *02:00* is received from SM_0). At the same time, the same event (e.g., the event with timestamp *01:00* from SM_0) can

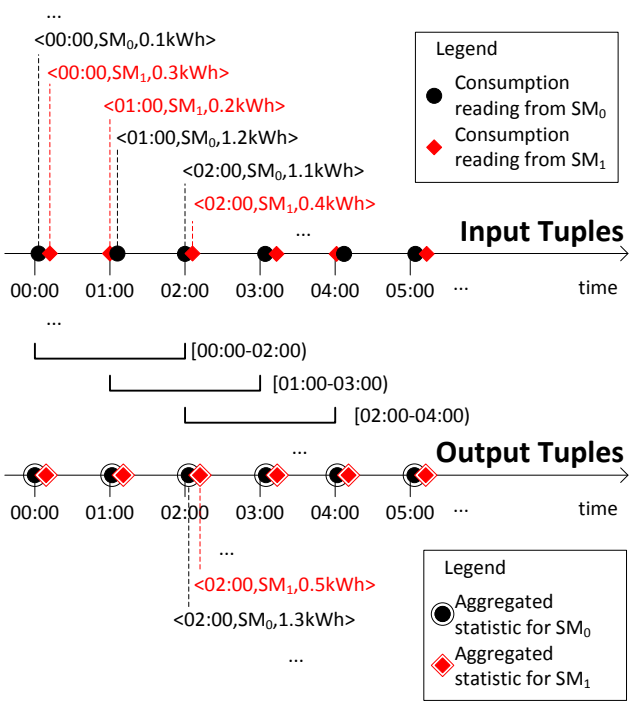


Figure 2: Sample aggregation of a stream of energy consumption readings reported by two SMs with a period *P* of one hour.

Parameter	Description
P	Events' polling (or pushing) period
WS	Aggregation window size
WA	Aggregation window advance
GB	Aggregation group-by attributes

Table 1: Parameters used to compute *Bes*' aggregated sums.

contribute to multiple sliding windows (*[00:00-02:00)* and *[01:00-03:00)* in the example).

2.2 Problem Statement

Bes' goal is to provide online privacy-preserving aggregation sums that prevent information leaks for the resulting statistics while complying with a given utility level.

In particular, given a set of smart meters $\text{SM}_1, \ldots, \text{SM}_n$ and $e_1^i, e_2^i, e_3^i, \ldots$ being the stream of events $\langle ts, sm, cons \rangle$ [2] generated by each SM_i , *Bes*' goal is to:

1. compute ϵ-*differentially private*[3] aggregated sums of the $cons$ readings given parameters *WS*, *WA*, *P* and *GB*,
2. maximize their resulting utility by minimizing the Mean Absolute Percentage Error (MAPE) [1], and
3. process such events in an online streaming-based fashion that can be carried out by resource-constrained devices (e.g., smart meters, Meter Concentrator Units or other inexpensive devices) which also allow for such online analysis to result in negligible deployment costs.

[2]Attribute ts represents the time when an event is generated and forwarded by the SM sm. Attribute $cons$ represents the consumption measured over a given period P.
[3]Defintion 2.1 defines ϵ-differential privacy.

As described in Section 1, statistics about energy consumption can disclose very sensitive information once shared with the public or third-party applications [16]. In the scope of a comprehensive privacy protection for AMIs' users, the guarantees of mechanisms preserving privacy through encryption (e.g., mechanisms encrypting the readings transmitted by users or homomorphic encryption schemes that could allow for such readings to be aggregated directly out of the ciphertext by untrusted applications) may not hold when aggregated data is intentionally shared with the rest of the world. For that reason, we build the protection in *Bes* on the differential privacy, as explained in the next section.

2.3 Differential Privacy

Differential privacy was introduced by Dwork et al. [8] to cover a specific problem: limiting privacy leaks of individual values when a statistical result is released. The released statistics should be more or less the same regardless whether a specific user (or value) partakes in the statistics, meaning that an adversary would not gain any more knowledge from the statistic including a specific value compared to one that does not include it. As we previously mentioned, this privacy dimension is crucial in AMI scenarios, and potentially overlooked in existing real-world privacy-protection schemes. The increasing interest in differential privacy is motivated by the proven guarantees of its original definition:

Definition 2.1. [9] A randomized function M gives ϵ-differential privacy for all data sets D and D' differing in at most 1 element, and all $S \subseteq \text{Range}(M)$, if

$$Pr[M(D) \in S] \leq exp(\epsilon) \times Pr[M(D') \in S],$$

where the probability is over the coin flips of M.

This definition states that a mechanism M enforces differential privacy if its result does not reveal whether M has been applied to D or D'. The parameter ϵ takes positive values and specifies *how close* to the original probability of outcomes the results obtained on the changed set should be. Smaller values of ϵ offer better differential privacy. A way for M to enforce differential privacy is given by:

Theorem 2.1. *[9] For $f : D \rightarrow R^d$, the mechanism M, which adds independently generated noise following the Laplace distribution $\mathcal{L}(\Delta f/\epsilon)$ to each of the d output terms, enjoys ϵ-differential privacy.*

The theorem states that differential privacy can be enforced if M adds calibrated noise to its result. Such noise depends on the function Δf (to be defined below), meaning that the price paid to enforce differential privacy (i.e., the added noise) is lower for some types of functions than for others. Intuitively, little noise is needed to prevent an adversary from discovering whether a given individual participated in a survey, since the total count of participants would only change by one (depending on whether the individual actually participated in the survey). On the other hand, more noise would be required for a differentially private sum of consumption readings, since each reading could be arbitrarily high. The *sensitivity* of a function is formally calculated as follows using the L_1 norm:

Definition 2.2. [9] For $f : D \rightarrow R^d$, the L_1 sensitivity of f is $\Delta f = max_{D,D'} \parallel f(D) - f(D') \parallel_1$ for all D, D' differing in at most 1 element.

That is, the sensitivity is 1 for a query that counts the number of elements in a dataset while it is equal to the highest possible

value of such elements for a query that sums them. When the same dataset can be queried multiple times, it is desirable for each query's outcome to enjoy ϵ-differential privacy. The possibility of sequentially combining two differentially private mechanisms is given in Theorem 2.2 [9].

Theorem 2.2. *[9] The composition of an ϵ_1-differentially private mechanism and an ϵ_2-differentially private mechanism is at worst $(\epsilon_1 + \epsilon_2)$-differentially private.*

This theorem implies that a mechanism running k queries on the same dataset and calibrating each query's noise to $\mathcal{L}(\Delta f/\epsilon)$ results in a $k\epsilon$-differentially private outcome. That is, noise drawn from the Laplace distribution $\mathcal{L}(k\Delta f/\epsilon)$ must be added to each of the k queries run by the mechanism in order for its outcome to enjoy ϵ-differential privacy. Nonetheless, this is not the case when such queries are posed to disjoint portions of the dataset. Theorem 2.3 by McSherry [20] (Theorem 4 in the original work) defines parallel composition and specifies that querying a dataset with disjoint queries does not incur a noise penalty. X denotes the input dataset and M_i the differentially private mechanism applied to each subsets D_i of the original domain D.

Theorem 2.3. *[20] Let M_i each provide ϵ-differential privacy. Let D_i be arbitrarily disjoint subsets of the input domain D. The sequence of $M_i(X \cap D_i)$ provides ϵ-differential privacy.*

With a mechanism satisfying Definition 2.1, it is possible to protect the data from very strong *adversaries*. That is, even when the adversary knows all but one of the points contributing to a query, the probability with which he can infer whether the missing point contributes to the query, and with what value, can be made arbitrarily low.

However, the protection against such a strong adversary comes at a price. The more differentially private the result, the more noise needs to be added, thus harming the usefulness of the result (which goes against one of our motivating challenges). In the next section we discuss how ϵ-differentially private sums are computed by *Bes*.

3. COMPUTING DIFFERENTIALLY PRIVATE AGGREGATED SUMS

This section presents *Bes*' differentially private aggregation mechanism and studies how its parameters affect the resulting utility.

As stated in [9], a given mechanism (e.g., that aggregates the events contributing to the same window) enjoys ϵ-differential privacy if it adds to its outcome noise drawn from the Laplacian distribution $\mathcal{L}(\Delta f/\epsilon)$, proportional to the global sensitivity of the function used to aggregate the events. When computing a sum, Δf refers to the highest observable value that contributes to the sum. That is, it does not depend on the particular events being aggregated but on the highest value that can be observed in the system (i.e., the sensitivity is not local to the events being aggregated). Given our system model, such a value exists and depends on the highest amount of energy E that can physically be consumed over a given period.[4] However (as later shown in Section 6), calibrating the noise over values substantially higher than the ones usually observed has a drastic effect on the accuracy of the summation.

Based on this observation, *Bes*' intuition is to calibrate Δf not on the maximum value that can be observed but rather on a lower bound *B*, closer to the energy consumption values usually reported. This implies that two features must be enforced in order for the aggregation to enjoy ϵ-differential privacy when Δf is set to *B*. On one hand, such a bound cannot depend on the particular events being aggregated but must be unique and shared for all the aggregations computed by *Bes* (Section 4 details how such a bound can be computed and shared). On the other hand, no event can contribute to the overall summation over a window with a consumption reading greater than *B*. Hence, given a window of events $e_1, \ldots, e_n$ such that $e_i.cons \in [0, E]$ and given a bound $B \in [0, E]$, rather than

$$S = \sum_{i=1}^{n} e_i.cons$$

we calculate

$$S_B = \sum_{i=1}^{n} \min(e_i.cons, B)$$

and subsequently add noise drawn from $\mathcal{L}(B/\epsilon)$.

Remark 1. By computing the latter, the error affecting the utility of the differentially private outcome now depends on two components, namely:

(i) the noise introduced to ensure differential privacy, refered to as Err_{noise}, and

(ii) the approximation error introduced by S_B, refered to as Err_{approx}.

Nonetheless, the overall error contribution of these components can be minimized depending on the bound *B*, the window size *WS*, the window advance *WA* and the period *P* (as discussed in Section 3.2).

3.1 Privacy preservation over sliding windows

As stated in Section 2.3, ϵ-differential privacy is provided by calibrating the noise to $\mathcal{L}(\Delta f/\epsilon)$ for two queries that cover disjoint portions of a dataset [20] while a noise drawn from the distribution $\mathcal{L}(k\Delta f/\epsilon)$ must be added when k queries are posed over the same dataset [9]. Due to *Bes*' continuous processing, each incoming event might contribute to more than one window, depending on how the latter evolves. Hence, the noise introduced by *Bes* must be calibrated correctly depending on the number of windows to which each incoming event contributes. Given parameters *WS* and *WA*, the number of windows to which each event contributes is $\lceil WS/WA \rceil$. We can thus state the following.

Claim 3.1. *Bes aggregation mechanism over a window of size* WS *and advance* WA *enjoys ϵ-differential privacy by adding noise drawn from the Laplace distribution $\mathcal{L}(k\mathrm{B}/\epsilon)$ to the bounded sum computed for each window, where $k = \lceil \mathrm{WS}/\mathrm{WA} \rceil$.*

3.2 Utility maximization

As stated in Section 2.2, we measure the utility of *Bes*' aggregated sums based on its MAPE. Such a value is computed as the mean of the absolute proportional distance between the exact and the differentially private sum for all the windows considered. For one window, the absolute percentage error is presented in Equation 1.

$$\mathrm{APE} = \left| \frac{S - (S_B + \mathcal{L}(kB/\epsilon))}{S} \right| = \left| \underbrace{\frac{S - S_B}{S}}_{Err_{approx}} - \underbrace{\frac{\mathcal{L}(kB/\epsilon)}{S}}_{Err_{noise}} \right| \quad (1)$$

Err_{approx} depends on the distance between the exact sum S and the bounded sum S_B while Err_{noise} depends both on the bound *B* and the exact sum S. Based on Equation 1, we discuss in the following how MAPE depends on the parameters *B*, *WS*, *WA*, *P*.

[4] E.g., 18.4 kWh for a single phase SM with an 80A fuse and 230V nominal voltage, considering a fully resistive load.

Influence of the bound B. The error introduced by Err_{approx} decreases for increasing values of *B*. More concretely, it ranges from 100% (for *B* = 0 kWh) to 0% (for *B* = E kWh). Contrary, the error introduced by Err_{noise} increases for increasing values of *B*. While the exact noise of each aggregated sum will vary when drawn from the Laplace distribution, the distribution is centered at 0 for *B* = 0 kWh and at $\mathcal{L}(kB/\epsilon)/S$ for *B* = E kWh. We can state that:

Remark 2. Given a window size *WS* and a period *P*, the minimum MAPE depends on how Err_{approx} decreases and Err_{noise} increases for increasing values of *B*.

Influence of the window size WS and window advance WA. Given that the Err_{noise} component of the MAPE depends on the exact sum S and given that the larger the window size, the higher the value of the resulting sum S is, we state the following remark.

Remark 3. For a given bound *B* and period *P*, the error introduced by the Err_{noise} component, and thus the MAPE of *Bes*' aggregation, is expected to decrease for increasing values of *WS* and *WA*.[5]

Influence of the period P. Given a bound *B* and window size *WS*, the number of events contributing to the same window and bounded to *B* depends on the period *P* under which they are measured. Since lower values of *P* would result in lower values for the consumption readings, they would also result in lower numbers of bounded consumption readings. Hence, the following remark holds.

Remark 4. For a given bound *B* and window size *WS*, the error introduced by the Err_{approx} component, and thus the MAPE of *Bes*' aggregation, decreases for decreasing values of the period *P*.

4. COMPUTING THE BOUND B

In order to comply with the same powerful adversary model as of the original definition of differential privacy, we cannot assume parameter *B* to be secret. This has an important consequence, as it implies that the method we use to compute *B* itself needs to be differentially private, in order to prevent an adversary in possession of *B* from inferring information about the dataset used to compute it. We outline in this section several techniques for choosing an appropriate value for the parameter *B* in practice.

We choose the bound *B* among a set of candidate values $\mathscr{B} = \{B_1, \ldots, B_o\}$ with a given mechanism M run over a dataset D_{explore}. D_{explore} contains the events used to quantify the utility of each individual bound in $\mathscr{B}$. We distinguish between D_{explore} and the set D_{live} of events over which bounded differentially private sums are later computed using the chosen *B* (we use the term *live* as such events are later processed in an online fashion), but we assume the points in both sets are drawn from the same underlying distribution.

To provide an example of why differential privacy is required for mechanism M, suppose $\mathscr{B}$ includes the median of D_{explore} and suppose *B* is chosen as such. Also, suppose an adversary is in possession of D'_{explore}, containing all except one of the events in D_{explore}. In such a case, if $B \notin D'_{\text{explore}}$, the adversary could learn the exact value of the missing point. Otherwise, the adversary could compare the chosen bound *B* with the one she herself can compute and learn if the missing point is larger or smaller than her calculated median, thus violating the privacy-preserving guarantees.

In the next section, we outline two different ways in which M, $\mathscr{B}$, *B* and D_{explore} can be restricted to ensure no information is leaked. First, we consider a scenario where D_{explore} either exists as an open dataset (or a dataset that could be released). Subsequently, we restrict the set of possible mechanisms to only be the ones that by themselves are differentially private, thus enabling us to also protect D_{explore}. We outline two such mechanisms and show how they compute *B*.

Utilizing open data repositories to compute B

A practical way to compute and release *B* is to rely on data from an already public D_{explore} dataset or from a dataset that can easily be made public. In many domains, such datasets would exist. In our particular context, we note that privacy concerns of energy consumption datasets decrease over time as energy consumption readings would refer to previous customers and the predictive power of individual points decreases as the corresponding individual habits change over time. Moreover, there exist several datasets that are collected and released on a voluntary basis. In our context, such a D_{explore} could indeed exist, as it might refer to past energy consumption readings, energy consumption readings referring to previous customers, energy consumption readings released on a voluntary-basis, and so on. Given a public D_{explore} dataset, any mechanism M and candidate set $\mathscr{B}$ based on that dataset can be chosen and used to compute *B*, as it would not increase the adversary's knowledge with respect to the already known D_{explore}. Nevertheless, datasets D_{explore} and D_{live} should be defined as disjoint datasets as D_{live} could not even be partially made public. *B* can be updated, for example, every year over a D_{explore} drawn from a set of public consumption readings measured during the previous year.

Choosing differentially private bounds B

A second option is to compute *B* as the result of differentially private queries over D_{explore}. While potentially resulting in a less accurate *B*, because of the noise introduced to preserve the privacy of the points in D_{explore}, such an approach can be leveraged even when a public D_{explore} dataset does not exist. Secondly, it allows for more frequent updates of the bound. Finally, it allows for D_{explore} and D_{live} to overlap. That is, since D_{explore} is not public, it can contain data that is later aggregated and released using the chosen bound *B*.[6]

To provide examples of such differentially private mechanisms, we outline two such methods below: *MCB* (Most Common B) and *HEB* (High Enough B). Both mechanisms leverage differentially private binary counting queries. These have the benefit of requiring only small amounts of noise to enjoy differential privacy as the sensitivity Δf of the counting mechanism is one. Thus they turn out to be quite accurate, as later shown in the evaluation in Section 6.

Method Most Common B (MCB). Given a set of smart meters $\text{SM}_1, \ldots, \text{SM}_n$ and $e_1^n, e_2^n, e_3^n, \ldots$ being the events of smart meter SM_n that contribute to D_{explore}, and given $\mathscr{B} = \{B_1, \ldots, B_o\}$, the method *MCB* aims at finding the bound $B \in \mathscr{B}$ resulting in the minimum MAPE for the majority of the SMs. Such a method could be leveraged when the optimal bound is close to the bound for which a dominant majority of SMs observes the minimum MAPE. Let MAPE_d^i be the MAPE for smart meter SM_d and tentative bound B_i. We define the help query function H_d (run against SM_d) as:

$$H_d(B_i, \mathscr{B}, D_{\text{explore}}) = \begin{cases} 1 & \text{if } \text{MAPE}_d^i < \text{MAPE}_d^j \\ & \forall B_j \in \mathscr{B} | B_j \neq B_i \\ 0 & \text{otherwise} \end{cases}$$

[5]We consider both the value of *WS* and *WA* to be increasing given that $k = \lceil WS/WA \rceil$.

[6]In this latter case, a higher amount of noise would be needed to enforce differential privacy, as stated in [9].

and the counting query C_i as:

$$C_i(B_i, \mathscr{B}, D_{\text{explore}}) = \sum_d H_d(B_i, \mathscr{B}, D_{\text{explore}})$$

In our implementation, queries $C_1, \ldots, C_o$ are run over D_{explore}, each query C_i counting the number of SMs for which B_i results in the minimum MAPE. Subsequently, the bound $B \in \mathscr{B}$ resulting in the highest count C_i is chosen as *B*. The events of each smart meter SM_d contribute to exactly one query C_i (since the minimum MAPE is observed by each smart meter for one bound B_i). As a result, disjoint portions of the dataset contribute to each query C_i and it is enough to add noise drawn from $\mathcal{L}(1/\epsilon)$ in order for query C_i to enjoy differential privacy [20]. Moreover, since *B* is subsequently chosen based on the outcomes of the differentially private queries $C_1, \ldots, C_o$, *B* is also differentially private [9].

Method High Enough B (HEB). Rather than focusing on the best bound for the majority of SMs in D_{explore}, the method *HEB* looks for the bound $B \in \mathscr{B}$ for which at least p of the n SMs observe a MAPE lower than the one observed for any higher bound B_j. Such a method could be leveraged when the optimal bound is greater than the one for which the majority of SMs observes the minimum MAPE. Given the help query function H^*_d (run against SM_d)

$$H^*_d(B_i, \mathscr{B}, D_{\text{explore}}) = \begin{cases} 1 & \text{if } \text{MAPE}^i_d < \text{MAPE}^j_d \\ & \forall B_j \in \mathscr{B} | B_j > B_i \\ 0 & \text{otherwise} \end{cases}$$

and the counting query C^*_i

$$C^*_i(B_i, \mathscr{B}, D_{\text{explore}}) = \sum_d H^*_d(B_i, \mathscr{B}, D_{\text{explore}})$$

this method queries D_{explore} and, for a given bound B_i, counts the number of SMs for which any bound smaller than or equal to B_i results in the minimum MAPE. The smallest bound B_i for which the count C^*_i is greater than or equal to $p * n$ is then chosen as *B*. Differently from mechanism *MCB*, the events of each smart meter SM_d potentially contribute to multiple queries C^*_i. For this reason, we add a higher amount of noise to the outcome of each query C^*_i. Such noise is drawn from $\mathcal{L}(o/\epsilon)$ if D_{explore} is queried o times, one for each possible bound in $\mathscr{B}$. On the other hand, based on the observation that

$$\sum_d H^*_d(B_i, \mathscr{B}, D_{\text{explore}}) \leq \sum_d H^*_d(B_j, \mathscr{B}, D_{\text{explore}}), \forall B_i < B_j$$

the noise can be lowered to $\mathcal{L}(\log_2(o)/\epsilon)$ by exploring $\mathscr{B}$ using a binary search method that only runs $\log_2(o)$ queries.

Both methods combine outcomes from differentially private mechanisms whose noise is calibrated accordingly to the number of times each event in D_{explore} is queried.

Claim 4.1. *Mechanisms* MCB *and* HEB *compute a bound* B *that enjoys ϵ-differential privacy and can thus be leveraged when a public dataset $D_{explore}$ is not available (or cannot be made public).*

We refer the reader to Section 6 for sample executions of methods *MCB* and *HEB* for a given set of candidate bounds $\mathscr{B}$ and dataset D_{explore}. Altogether, Remarks 2, 3, 4 and Claims 3.1, 4.1 address the first and second challenges stated in Section 2.2. We discuss the third challenge in the following.

5. DISTRIBUTED EVENT AGGREGATION

Sections 3 and 4 discussed how *Bes* addresses two of its motivating challenges, namely privacy preservation and utility maximization. The third challenge *Bes* aims to solve is scalable and distributed online event aggregation. That is, we aim for an efficient implementation of *Bes* that can be run by the heterogeneous embedded devices composing the advanced metering infrastructure, enabling for event aggregation to happen closer to the sources (i.e., processing smart meters' events in a distributed fashion) and for live information to be leveraged in applications such as fraud detection or energy consumption forecast ones. In this section, we present *Bes* implementation, discussing its complexity (complementing the empirical evaluation later presented in Section 6.3) and provide an example of its execution for a sample sequence of input events.

Implementation. As introduced in Section 2.1, we assume that the events carrying the energy consumption readings are fed in timestamp order to the instance of *Bes* that processes them. Moreover, if such consumption readings are collected from multiple input streams, they are first merge-sorted and subsequently processed as a single stream of timestamp-sorted events (as discussed in [5, 15]) in order for *Bes* to aggregate them deterministically.

Algorithm 1 presents *Bes* implementation[7]. Instances of the class `Window` (L. 1-13) maintain the bounded sum of attribute A_i for a given window and value of the *GB* parameter. Variables `ts`, `gb`, `bsum`, `B`, `k` and `epsilon` (L. 2-7) represent the window's start timestamp, its *GB* value, the bounded sum and the parameters *B*, k and ϵ, respectively. Function `add` (L. 9) is invoked to update the bounded sum, increasing `bsum` by the lower value among the event's attribute A_i and the bound B, for each event contributing to the window. Function `getOutput` (L. 12) is invoked to retrieve the differentially private sum after the Laplacian noise (drawn from the distribution presented in Claim 3.1) is computed with the function `lapl` (not shown in the code) and added to the variable `bsum`. Output events are composed by three attributes, carrying the start timestamp of the window to which the contributing events refer, the *GB* value and the bounded differentially private sum.

The `Bes` class (L. 15-43) defines the processing steps executed for each new incoming event e. The `windows` variable (L. 19) maintains all the active windows for the different values of the *GB* parameter. Upon reception of a new incoming event e, the `process` function (L. 21) is invoked to update the windows maintained by *Bes*. This function starts by extracting the timestamp and the *GB* value (an empty string if the latter is not defined) of e (L. 22-24). Subsequently, function `getTargetWindows` (not shown in the code) is invoked to retrieve the start timestamps of all the windows to which e contributes depending on the *WS* and *WA* parameters. The execution continues by iterating in timestamp order through the `windows` maintained by *Bes* and by producing the output events of the windows to which e (and future incoming events too) no longer contribute. That is, an output event is produced for each window whose starting timestamp is lower than the earliest among the ones in `windowsTS` and the window is then removed (L. 31-34). Finally, all windows to which e contributes are created (if e is their first contributing event) and updated (L.38-43).

Complexity analysis. Based on the proposed implementation, the following claim holds:

[7]The algorithm presents a simplified version of the implementation available at https://github.com/vincenzo-gulisano/Bes.

Algorithm 1: Window class, computing the bounded differentially private sum of a generic attribute A_i for a given window of size *WS* and advance *WA* and for a given *GB* parameter, and *Bes* class, defining the processing steps executed for each new incoming event e.

```
1  class Window
2    int ts // start timestamp
3    string gb // group-by
4    int bsum // bounded sum
5    int B
6    int k // ⌈WS/WA⌉
7    int epsilon
8
9    void add(Event e)
10     bsum += min(e.A_i, B)
11
12   Event getOutput()
13     return Event(ts,gb,bsum+lapl(k*B/epsilon))
14
15 class Bes
16   int WS // window size
17   int WA // window advance
18   int earliest_ts // earliest timestamp
19   map<int,<map<string,Window>>> windows
20
21   void process(Event e) // upon reception of event e
22     int ts = e.ts
23     // Get GB value (empty string if GB is not defined)
24     string gb = groupby(e,GB)
25
26     // Get start timestamps of windows
27     // to which e contributes
28     list<int> windowsTS = getTargetWindows(ts,WS,WA)
29
30     // produce result and purge stale windows
31     while (earliest_ts < windowsTS.front())
32       wins = windows.get(earliest_ts)
33       for (Window w : wins)
34         output(w.getOutput())
35       earliest_ts += WA
36
37     // update active windows
38     for (winTS : windowsTS)
39       if(!windows.contains(winTS))
40         windows.insert(winTS,new map<string,Window>())
41       if(!windows.get(winTS).contains(gb))
42         windows.get(winTS).put(gb,new Window())
43       win.get(winTS).get(gb).add(e)
```

Claim 5.1. *Algorithm 1 computes differentially private aggregation sums given parameters* B*, k,* ϵ*,* GB*,* WS*, and* WA *with space and time complexities linear in the number of active windows* $\lceil$WS/WA$\rceil$ *and the number of distinct* GB *values observed for such windows.*

It should be noted that the `Window` class incurs constant space complexity. At the same time, its functions have constant time complexity. As stated in Claim 5.1, the number of windows maintained by *Bes* depends on the aggregation parameters *GB*, *WS*, and *WA*. More concretely, its depends on $\lceil WS/WA \rceil$ (as discussed in Section 3) and on the schema attribute(s) chosen for the *GB* parameter. These dependencies can be leveraged to decrease the space and time requirements of *Bes*. For instance, by not defining a *GB* parameter, exactly one window would be maintained for each given starting timestamp. Furthermore, the number of distinct windows could also be lowered by aggregating energy consumption readings from households belonging to the same district. In such a case, the number of distinct *GB* values would be lower than the one incurred by aggregating each household's readings individually.

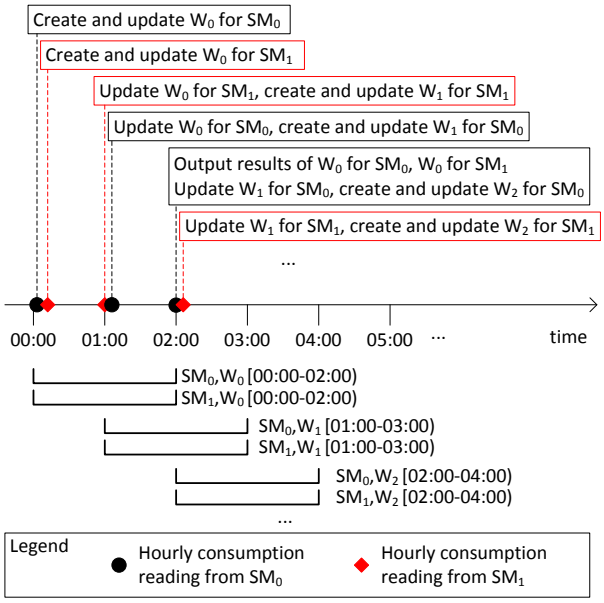


Figure 3: Sample execution of *Bes* for a stream of energy consumption readings reported by two SMs with a period *P* of one hour.

Sample execution. Reprising the example introduced in Section 2.1, Figure 3 presents the sample stream of energy consumption readings reported by two SMs (with a periodicity *P* of one hour) and the processing steps for an instance of *Bes* computing the bounded differentially private sum of each SM (given a window size *WS* of two hours and a window advance *WA* of one hour).

Upon reception of an incoming event, new windows are created and, together with existing ones, updated (e.g., when processing the event from SM_0 at time 01:00). At the same time, all the output events for the windows sharing a given start timestamp are produced (in timestamp-order) upon reception of the first event that does not contribute to any of them any longer (e.g., when processing the event from SM_0 at time 02:00). As shown in Figure 3, readings sharing the same timestamp but generated by different SMs might be arbitrarily interleaved because of their inter-arrival times. Nonetheless, all the output events for the windows sharing the same start timestamp are created deterministically upon reception of the first event not contributing to any of them.

6. EVALUATION

This section covers the evaluation of *Bes*. First, we present how the bound *B*, the window size *WS* and the period *P* can be chosen to maximize the utility (i.e., to minimize the MAPE), as discussed in Section 3. Subsequently, we compare the resulting MAPE for a D_{live} dataset given the bound *B* chosen using the *MCB* and *HEB* methods (discussed in Section 4) and a D_{explore} dataset. Finally, we evaluate the performance of *Bes* (in terms of processing throughput and latency) and further discuss its leveraging in AMIs applications. We begin by introducing the data, the software and hardware setup and we conclude with a summary of our findings.

Data overview. The data used in this evaluation consists of per-appliance energy consumption readings (measured in kWh) collected from 308 Swedish households from January 2006 to July 2008. Such readings are taken with a period *P* of 10 minutes. This data contains a small set of potentially incorrect values (e.g., suspi-

Label	*GB*	*WS*	*P*	*B* (kWh)	*Dataset*
Q_1	sm	24 h	1 h	$0, 1, \ldots, 15$	D_{explore}
Q_2	sm	48 h	1 h	$0, 1, \ldots, 15$	D_{explore}
Q_3	sm	72 h	1 h	$0, 1, \ldots, 15$	D_{explore}
Q_4	sm	96 h	1 h	$0, 1, \ldots, 15$	D_{explore}
Q_5	sm	24 h	30 m	$0, 1, \ldots, 15$	D_{explore}
Q_6	sm	24 h	10 m	$0, 1, \ldots, 15$	D_{explore}
Q_7	sm	24, 48, 72, 96 h	1 h	3	D_{live}
Q_8	sm	24, 48, 72, 96 h	1 h	7	D_{live}

Table 2: Labels and parameters of the evaluation queries.

ciously high consumption values) and misses energy consumption readings over certain periods of time (distinct for each household). Despite its noisy and lossy nature, in our evaluation we opted for using the data as it is (i.e., without sanitizing it) in order for results to resemble the ones observed in real-world applications (where pre-processing or sanitization schemes might not be in place).

According to the discussion in Section 4, data has been divided into two sets D_{explore} and D_{live}. Dataset D_{explore} contains the data measured from January 2006 to December 2007 (included) and is used to explore the MAPE for different *B*, *WS* and *P* values. Dataset D_{live} contains the data measured from January 2008 to July 2008 (included) and is used to compare the MAPE after choosing *B* with the *MCB* and *HEB* methods applied to D_{explore}.

In the following experiments, data has been pre-aggregated on a per-household basis (i.e., aggregating consumption readings referring to distinct appliances belonging to the same household). When considering a period *P* different than 10 minutes, the data is also pre-aggregated (before being fed to *Bes*) over a sliding window with *WS* and *WA* set to *P* (i.e., if *P*=1 hour, all the consumption readings referring to the same hour are pre-aggregated together). Notice that this does not alter the correctness of the input data (nor the outcome). That is, given a period *P* of 1 hour, this pre-aggregation produces the energy consumption readings that would have been collected from the households every hour.

Table 2 presents the labels and the parameters of the evaluation queries. The different experiments are designed in order to show the effect of each parameter to the resulting aggregation utility of each window. As stated in Claim 3.1, the noise added by *Bes* to the aggregated sums of a window is drawn from the $\mathcal{L}(kB/\epsilon)$ distribution.

Since the contribution of k and ϵ to the resulting utility is known (the higher k and the lower ϵ, the higher the noise), we assume, without loss of generality, both k and ϵ are 1 in the rest of this evaluation ($k = 1$ implies *WS* and *WA* are increased accordingly).

Hardware and software setup. *Bes* has been implemented on top of Storm [26]. Our setup consists of two machines. A server equipped with a 2.6 GHz AMD Opteron 6230 (48 cores over 4 sockets) with 64 GB of memory and a single-board deviced called Odroid-XU3 [22] (or simply Odroid in the remainder), equipped with a Samsung Exynos5422 Cortex-A15 2.0 GHz quad core and Cortex-A7 quad core Mali-T628 MP6 and with 2 GB of memory. In the setup, the Odroid runs the Storm topology implementing *Bes* while the Server runs Storm's coordinator and also maintains *Bes*' input and output queues. As discussed in Section 1, we aim at an online execution of *Bes*' analysis in distributed many-core systems. For this reason, we conducted our evaluation with a single-board device such as the Odroid. Despite being more powerful than an existing smart meter, a dedicated network of such devices (smaller in size than the network of smart meters) could be deployed in AMIs not only to measure and collect data, as done in existing setups with meter concentrator units, but also to process the latter in a real-time fashion, while resulting in negligible deployment costs for energy providers willing to run *Bes* in a distributed and parallel fashion.

6.1 Utility Maximization

This section studies how parameters *B*, *WS* and *P* can be chosen to minimize the aggregation's MAPE (thus maximizing its utility).

Influence of bound *B*. This experiment runs query Q_1 (Table 2), computing the aggregated sum (on a per-household basis) over windows of 24 hours. The experiment measures the MAPE and its $\text{Err}_{\text{approx}}$ and $\text{Err}_{\text{noise}}$ (discussed in Section 3) for 16 different values of *B* (from 0 kWh up to 15 kWh). The highest value for *B* (15 kWh) has been chosen according to [23], stating that such a value represents the average per capita energy consumption for the households' country during the years 2006 and 2007. The candidate set can be created in a similar manner as it is presented in [7].

Results are presented in Figure 4a. As discussed in Section 3, the $\text{Err}_{\text{approx}}$ measures 100% for *B*=0 kWh while it decreases for larger values of *B*. At the same time, the $\text{Err}_{\text{noise}}$ grows from 0% (for *B* = 0 kWh) to approximately 75% (for *B* = 15 kWh). The lowest MAPE (approximately 25%) is achieved by the optimal bound *B*=3 kWh.

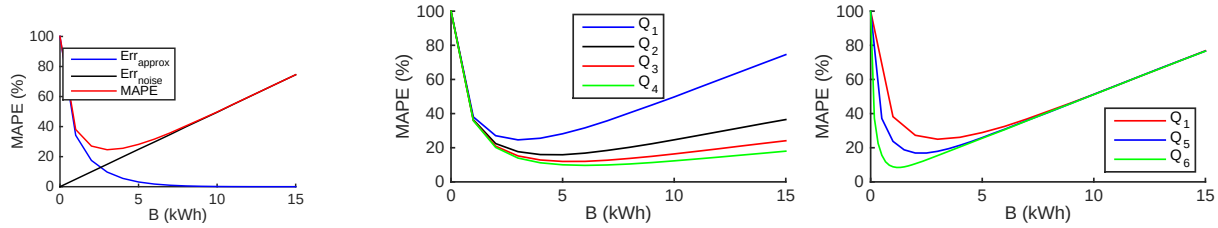
Influence of window size *WS*. In this experiment, we study how the MAPE evolves for queries Q_1, Q_2, Q_3 and Q_4 (which differ in their *WS* parameter). As presented in Figure 4b, in all the experiments the same behavior for the MAPE can be observed for an increasing bound *B*. As it can be noticed, the optimal bound *B* changes for different values of *WS* (e.g., 3 kWh for a window size of 24 hours versus 6 kWh for a window size of 96 hours). Moreover, different minimum MAPE are observed for different window sizes (e.g., approximately 25% for a window size of 24 hours versus 10% for a window size of 96 hours). Finally, a decreasing improvement is also observed for increasing window sizes. For instance, an improvement of approximately 36% is observed for a window size of 48 hours rather than 24 hours, while an improvement of 25% is observed for a window size of 72 hours rather than 48 hours.

Influence of period *P*. In this experiment, we consider three different *P* values (queries Q_1, Q_5 and Q_6) and study how the MAPE evolves for increasing values of *B*. As discussed in Section 3, given the same bound *B*, a lower *P* results in a lower value for the $\text{Err}_{\text{noise}}$, and thus in a lower MAPE. This can be observed in Figure 4c. The minimum MAPE for *P* values of 1 hour, 30 minutes and 10 minutes is measured at 25%, 17% and 8.5%, respectively.

6.2 MAPE for MCB and HEB methods

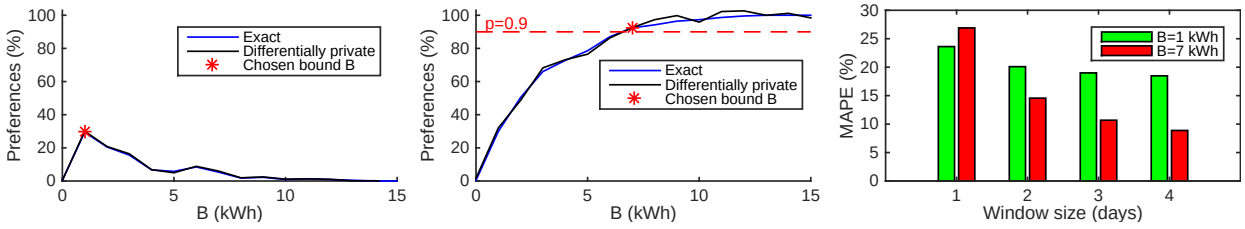
In this section, we present the MAPE observed for the D_{live} dataset for two different bound values computed applying methods *MCB* and *HEB* to the D_{explore} given $\mathscr{B} = \{0 \text{ kWh}, \ldots, 15 \text{ kWh}\}$, as discussed in Section 4.

Figure 5a presents the exact and differentially private answers for an execution of the *MCB* queries for the D_{explore} dataset and the given $\mathscr{B}$ set. As it can be observed, the difference between the exact and differentially private answers is negligible (as discussed in Section 4, the *MCB* method requires the addition of noise drawn from $\mathcal{L}(1)$ in order to preserve the privacy of the queries' outcomes). Based on both the exact and differentially private answers, the resulting bound *B* is 1 kWh. Figure 5b presents the exact and differentially private answers for the *HEB* method and p set to 0.9. For method *HEB*, the noise required to preserve the queries' privacy is proportional to $\log_2 o$, where o is the number of candidate bound values. This implies noise must be drawn from $\mathcal{L}(4)$ to



(a) $\text{Err}_{\text{approx}}$, $\text{Err}_{\text{noise}}$ and MAPE for query Q_1. (b) MAPE for queries Q_1, Q_2, Q_3 and Q_4. (c) MAPE for queries Q_1, Q_5 and Q_6.

Figure 4: Evaluation - Utility maximization experiments



(a) Results for *MCB* over D_{explore} dataset. (b) Results for *HEB* over D_{explore} dataset. (c) MAPE for queries Q_7 and Q_8.

Figure 5: Evaluation - *MCB* and *HEB* methods.

preserve the queries' outcomes for the given S_B set. As observed in the figure, the difference between the exact and differentially private outcome of the queries is higher than the one observed in the previous case. Nonetheless, based both on the exact and differentially private answers, the resulting bound *B* is 7 kWh.

Figure 5c presents the resulting MAPE for the chosen bounds and queries Q_7 and Q_8. As presented in the figure, the MAPE behavior observed for the D_{explore} dataset is also found for the D_{live} dataset. Both for *B* of 1 kWh and 7 kWh, a decreasing MAPE can be observed for increasing window sizes. Moreover, while resulting in an higher MAPE for a window of 24 hours, the bound of 7 kWh results in a lower MAPE for increasing window sizes.

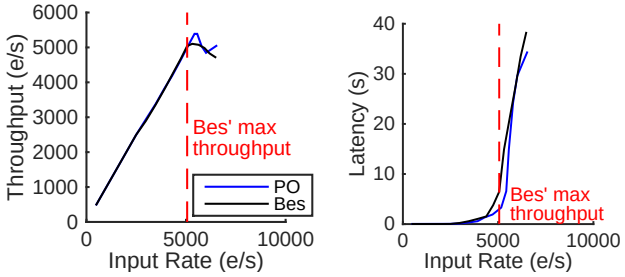
6.3 Performance Evaluation

In this section, we run query Q_6 and, without altering the events being fed to *Bes*, we change the rate at which such events are injected, studying *Bes*' performance in terms of throughput and latency. For completeness, we also measure the throughput and latency of a Privacy-Oblivious aggregate (referred to as PO), which does not provide differential privacy.

The throughput is measured as the number of events consumed per second (e/s). The latency represents the timestamp difference (in seconds) between an output event and the latest input event that produced it.

Figure 6a shows how *Bes*' and PO's throughput curves evolve for an increasing input rate. As it can observed, both curves increase linearly with the input rate up to the maximum throughput, measured at 5,100 e/s for *Bes* and 5,400 e/s for PO. On one hand, this result indicates that the computational overhead introduced by *Bes* to provide differentially private aggregation is small, resulting in a throughput degradation (compared with PO) lower than 6%. On the other hand, it also indicates that, for a period *P* of 10 minutes and the given query Q_6, a single Odroid device can potentially aggregate the readings forwarded by approximately 3 million smart meters.

Figure 6b shows how the latency curves evolve accordingly to the increasing rates. As it can be observed, the two curves evolve similarly (as reasonably expected, PO results in a lower latency).



(a) Throughput evolution. (b) Latency evolution.

Figure 6: Throughput and latency evolution for query Q_6.

For *Bes*, the latency when the input rate does not exceed 2,000 e/s is measured at approximately 80 milliseconds. As *Bes* gets closer to its saturation, the latency starts growing with a harsh slope up to approximately 8 seconds for a throughput of 5,100 e/s.

Leveraging Bes in AMIs applications. As mentioned in Section 1, fraud detection and energy consumption forecast represent possible use-cases for *Bes*. For the former, the error introduced by *Bes* (as low as 9%) would not undermine fraud detection schemes which, for a given household, consider changes in its average energy consumption greater than 50% to be suspicious [3]. At the same time, *Bes* would allow for a single Odroid device to aggregate 5,100 e/s with a latency (8 seconds) that is negligible with respect to the time (10 to 30 minutes [27]) available to forecast energy consumption.

Highlights of the evaluation

As discussed, the utility for the differentially private sum aggregation can indeed be maximized by tuning its parameters *B*, *WS* and *P*. The evaluation has also shown that results observed for the D_{explore} dataset are also observed for the D_{live} dataset given the bound released by the *MCB* and *HEB* methods. Finally, we have shown how off-the-shelf, cheap single-board devices are able to meet a high processing throughput (approximately 5,100 e/s) and,

once deployed in a real-world advanced metering infrastructure, able to process the data of potentially millions of SMs while resulting in negligible deployment costs for the energy provider maintaining the advanced metering infrastructure.

7. RELATED WORK

Even though the work in this paper is focused on cyber-physical systems such as AMIs, scenarios in need of joint addressing of computational and privacy challenges are flourishing in other domains also, as evidenced by research that has focused on (continuous) analysis of evolutionary streams of binary data [9], web data [6, 19], chemical and biological compounds [21], vehicular traffic data [11, 19, 24] and salary data [4]. In the following, we discuss related work focusing on distributed analysis and privacy preservation in AMIs, the main focus of this paper.

Data streaming has shown to be a good candidate for scalable and efficient distributed processing of AMIs' events. Among others, streaming-based analysis has been studied for intrusion detection [10, 13] and data validation [14]. To the best of our knowledge, *Bes* is the first framework that takes a system perspective to (i) study and identify the trade-offs in differentially private data streaming aggregation and (ii) embrace both the differential privacy and streaming aggregation challenges while enabling the use of inexpensive single-board devices.

Different privacy issues in AMIs have been studied in the recent years and for an overview of the different privacy enhancing methods proposed so far we refer the reader to the work of Jawurek et al. [17] and Tudor et al. [28]. In the following we focus on related work which employs differential privacy in AMIs. Mainly, related work differs from *Bes* because (1) it modifies/weakens the original adversary model introduced by differential privacy; and/or (2) it has been conducted (or simulated) with synthetic data; and/or (3) it does not focus on aggregated sums. Barthe et al. [2] introduce a protocol able to aggregate smart meter readings in a privacy-friendly fashion by means of distributed noise generation but do not evaluate it based on real events. Aggregation of smart meters' data is also discussed by Ács and Castelluccia [1]. Also in this case, experiments are based on synthetic data. Moreover, the bound B is assumed to be known a priori (as we discussed in this paper, choosing a conservative bound B such as the highest consumption that can be measured over P can result in drastic accuracy degradation for the resulting). Shi et al. [25] discuss an orthogonal problem (the presence of untrusted aggregators) which is solved by leveraging homomorphic encryption. Differently from us, their evaluation focuses on streams of binary data and is tailored for a weaker adversary than the original adversary introduced in [8]. Zhao et al. [29] present an alternative privacy enhancing technology based on BLH (Battery-based Load Hiding) and differential privacy. Nonetheless, their method can only preserve the privacy of appliances consuming less than 200W (*Bes* does not impose restrictions about the appliances behind the events). Finally, Jelasity et al. [18] assume the existence of a bound in order to limit the global sensitivity of the aggregation and discuss a method to prevent differentially private queries run continuously over time from enabling the adversary to learn the readings' underlying distribution. The mechanism is orthogonal to the underlying differentially private scheme that, such as *Bes*, enforces differential privacy for the individual statistics to be released. We further complement their work by presenting several techniques which can be used to compute such a bound in a differentially-private fashion. Day and Li [7] propose DP-Sense, an algorithm which is employed to release noisy column counts in high-dimensional binary datasets. The algorithm probabilistically chooses a sensitivity threshold θ which limits each row contribution to the column count, reducing the noise added to the count and improving the utility. The results presented in their study are obtained only on binary datasets, but it is mentioned that in the case of non-binary datasets the magnitude of noise is larger and it is given by a maximum value r contained in the database. Our mechanisms for obtaining the bound B (*MCB* and *HEB*) are orthogonal to theirs and are adapted to the problem of obtaining AMI data DP-sums. However, further investigation in how to best design and evaluate a framework to choose the bound would be interesting. Fan and Xiong [12] also describe an interesting complimentary method. For now their method is focused on counting queries but the authors claim it can also be extended to sums in future work (the topic of this paper).

As discussed by Kellaris at al. [19], the privacy granularity of a differentially private mechanism can protect *events* (e.g., a specific reading among the readings of a SM), *users* (e.g., the reading of a user among the ones of a set of users) or sequences of contiguous events occurring in successive time instants. While aggregation parameters can be chosen to enforce a desired level of protection, we show with *Bes* they can also be chosen to maximize the analysis utility, independently of the overall number of queries posed over the data, which might depend on privacy budget allocations [19] or complementary privacy-preserving schemes such as [18].

8. CONCLUSIONS

In this paper, we take a system perspective and present *Bes*, whose goal is the aggregation of energy consumption readings in a privacy-preserving, efficient and distributed fashion. Focusing on cyber-physical systems, and especially on advanced metering infrastructures, we studied how privacy-preserving calibrated noise can be added to aggregated energy consumption measurements in order for them to be differentially private. At the same time, we showed how the error introduced by such noise can be minimized by limiting the contribution of each measurement to a given bound B and by tuning the aggregation itself. We evaluated the proposed method with events collected from a real-world advanced metering infrastructure and showed that *Bes* can achieve errors lower than 10%. At the same time, we showed how thousands of events per second can be aggregated with low processing latency by inexpensive single-board devices representative of the ones deployed in many large scale infrastructures.

Bes defines a major step forward towards the adoption of differential privacy in real-world applications. Future directions for *Bes* include the extension of its statistical analysis capabilities to other large scale infrastructures and other statistical functions. At the same time, it would be of interest to explore possible approximated results achieved by leveraging counting queries, as done in this paper to compute the bound B.

Acknowledgments

The research leading to these results has been partially supported by Chalmers Transportation Area of Advance, by the collaboration framework of Chalmers Energy Area of Advance project "SN09: EXAMINE", by the Swedish Civil Contingencies Agency (MSB) through the project "RICS", by the Swedish Foundation for Strategic Research (framework project FiC) and with support from the Swedish Energy Agency under the program Energy, IT and Design.

9. REFERENCES

[1] G. Ács and C. Castelluccia. I have a DREAM!: Differentially private smart metering. In *Proceedings of the 13th International Conference on Information Hiding*, IH'11. Springer-Verlag, 2011.

[2] G. Barthe, G. Danezis, B. Gregoire, C. Kunz, and S. Zanella-Beguelin. Verified computational differential privacy with applications to smart metering. In *Computer Security Foundations Symposium (CSF), 2013 IEEE 26th*, June 2013.

[3] J. E. Cabral, J. O. Pinto, and A. M. Pinto. Fraud detection system for high and low voltage electricity consumers based on data mining. In *Power & Energy Society General Meeting. PES'09. IEEE*, 2009.

[4] J. Cao, Q. Xiao, G. Ghinita, N. Li, E. Bertino, and K.-L. Tan. Efficient and accurate strategies for differentially-private sliding window queries. *Cyber Center Publications*, Jan. 2013.

[5] D. Cederman, V. Gulisano, Y. Nikolakopoulos, M. Papatriantafilou, and P. Tsigas. Concurrent data structures for efficient streaming aggregation. *Report, Chalmers University of Technology*, 2013.

[6] T. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. In *Automata, Languages and Programming*. Springer, 2010.

[7] W.-Y. Day and N. Li. Differentially private publishing of high-dimensional data using sensitivity control. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 451–462. ACM, 2015.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In S. Halevi and T. Rabin, editors, *Theory of Cryptography*, number 3876 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2006.

[9] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the ACM Symposium on Theory of Computing*, STOC '10. ACM, 2010.

[10] M. A. Faisal, Z. Aung, J. R. Williams, and A. Sanchez. Securing advanced metering infrastructure using intrusion detection system with data stream mining. In *Intelligence and Security Informatics*. Springer, 2012.

[11] L. Fan and L. Xiong. Real-time aggregate monitoring with differential privacy. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12. ACM, 2012.

[12] L. Fan and L. Xiong. An adaptive approach to real-time aggregate monitoring with differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2094–2106, Sept 2014.

[13] V. Gulisano, M. Almgren, and M. Papatriantafilou. METIS: a two-tier intrusion detection system for advanced metering infrastructures. In *Proceedings of the 5th international conference on Future energy systems*. ACM, 2014.

[14] V. Gulisano, M. Almgren, and M. Papatriantafilou. Online and scalable data validation in advanced metering infrastructures. In *The 5th IEEE PES Innovative Smart Grid Technologies (ISGT) European Conference*, 2014.

[15] V. Gulisano, Y. Nikolakopoulos, M. Papatriantafilou, and P. Tsigas. Scalejoin: A deterministic, disjoint-parallel and skew-resilient stream join. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 144–153, 2015.

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

[17] M. Jawurek, F. Kerschbaum, and G. Danezis. Sok: Privacy technologies for smart grids–a survey of options. *Microsoft Res., Cambridge, UK*, 2012.

[18] M. Jelasity and K. P. Birman. Distributional differential privacy for large-scale smart metering. In *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, pages 141–146. ACM, 2014.

[19] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12), 2014.

[20] F. McSherry. Privacy integrated queries. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD)*. Association for Computing Machinery, Inc., June 2009.

[21] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler. Gupt: Privacy preserving data analysis made easy. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12. ACM, 2012.

[22] Odroid-XU3. http://www.hardkernel.com.

[23] The World bank. Electric power consumption (kWh per capita). http://data.worldbank.org/indicator/EG.USE.ELEC.KH.PC?page=1.

[24] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10. ACM, 2010.

[25] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *NDSS*, volume 2, 2011.

[26] Storm project. http://storm.apache.org/.

[27] J. W. Taylor. An evaluation of methods for very short-term load forecasting using minute-by-minute british data. *International Journal of Forecasting*, 2008.

[28] V. Tudor, M. Almgren, and M. Papatriantafilou. Analysis of the impact of data granularity on privacy for the smart grid. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, pages 61–70, New York, NY, USA, 2013.

[29] J. Zhao, T. Jung, Y. Wang, and X. Li. Achieving differential privacy of data disclosure in the smart grid. In *INFOCOM, 2014 Proceedings IEEE*, April 2014.