2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture

Guaranteeing Local Differential Privacy on Ultra-low-power Systems

Woo-Seok Choi, Matthew Tomei, Jose Rodrigo Sanchez Vicarte, Pavan Kumar Hanumolu, Rakesh Kumar
University of Illinois
Urbana, IL, USA
Email: {wchoi33, tomei2, joser2, hanumolu, rakeshk}@illinois.edu

***Abstract*—Sensors in mobile devices and IoT systems increasingly generate data that may contain private information of individuals. Generally, users of such systems are willing to share their data for public and personal benefit as long as their private information is not revealed. A fundamental challenge lies in designing systems and data processing techniques for obtaining meaningful information from sensor data, while maintaining the privacy of the data and individuals. In this work, we explore the feasibility of providing local differential privacy on ultra-low-power systems that power many sensor and IoT applications. We show that low resolution and fixed point nature of ultra-low-power implementations prevent privacy guarantees from being provided due to low quality noising. We present techniques, *resampling* and *thresholding*, to overcome this limitation. The techniques, along with a privacy budget control algorithm, are implemented in hardware to provide privacy guarantees with high integrity. We show that our hardware implementation, DP-Box, has low overhead and provides high utility, while guaranteeing local differential privacy, for a range of sensor/IoT benchmarks.**

***Keywords*-differential privacy; randomized response; RAPPOR; microcontrollers; low-power systems; IoT**

I. Introduction

With the rapid technology development of low-power sensors and computing systems, sensor networks or Internet of Things (IoT) are becoming increasingly wide-spread and power a large variety of applications, including, but not limited to, wearables [1], implantables [2], environmental, health, and structural monitors [3], and mobile systems [4]. All trends [5] and predictions [6] point to further ubiquity of these sensor and IoT systems in the future.

Unfortunately, sensors in mobile devices or IoT systems generate data that may contain private information of individuals. Data on medical health [7], location [8], energy consumption [9], and personal preferences [10], for example, are routinely transmitted by such systems to the cloud or peers.

Generally, users of mobile or IoT systems are willing to share their data for public benefit (and, ultimately, their own benefit), but at the same time they do not want their private information to be revealed [11]. For example, a user may seek to benefit from the recipes designed by a web-based weight loss program without having to reveal one's true weight. Another user may seek to benefit from the traffic alerts generated by a web-based alert system [12] without revealing one's true location. Yet another user may be willing to help a cloud-based learning model [13] train, as long as one's true data on home energy consumption or thermostat preferences are not revealed. A fundamental challenge lies in designing systems and data processing techniques for obtaining meaningful information from sensor data, while maintaining the privacy of the data and individuals.

Differential privacy [14] is an approach to providing data privacy that has gained significant attention over the past few years in the data-mining and machine-learning communities. It can used to gain statistical information from large-scale database [15] while ensuring individual privacy by adding properly scaled and distributed noise to the statistical query output (Section II-A). Conventional differential privacy (DP) mechanisms assume that there is a trusted database server collecting all the personal data. However, there are many potential attacks that can reveal the private data before they reach the trusted party [16]. This is especially a big issue in sensor networks or IoT systems. For such systems, it is also possible to apply DP in the local setting (or *Local Differential Privacy*), where there is no trusted data curator, and each sensor submits the private data after adding noise individually (Section II-B). In this case, when others receive noised output from a sensor, they should be unable to tell the original sensor data because all the possible sensor data have similar probabilities to report the same noised output.

In this work, we explore the feasibility of providing local differential privacy (LDP) on ultra-low-power (ULP) systems that power many sensor and IoT system applications. ULP processors are already the mostly widely used processors today. In fact, due to their ubiquity in sensor and IoT systems, their production already far exceeds personal computers and mobile processors [17]. The 2015 ITRS report [18] projects that sensor and IoT applications will continue to rely on such processors in the future. Sensors often appear as co-processors or peripherals in these processors.

We find that naive hardware implementation of differential privacy mechanism cannot guarantee local privacy of individual sensors in ULP systems (Section III-A3). Energy constraints require ULP processors and sensors to support low resolution, fixed point hardware. We show that such hardware cannot generate the quality of noise needed to provide local differential privacy guarantees. In order to overcome this limitation, we propose new techniques - *resampling* and *thresholding* that can ensure local privacy in spite of low resolution and fixed point hardware (Section III-B). We also propose a privacy budget control technique for such hardware that exploits thresholding and resampling

2575-713X/18/$31.00 ©2018 IEEE
DOI 10.1109/ISCA.2018.00053

to reduce privacy loss (Section III-C). We then implement these techniques in a hardware module, called DP-Box (Section IV) - DP-Box can either be part of the processor or the sensor controller and is responsible for noising sensor data in a way that guarantees local privacy with high integrity. Finally, we discuss the latency and utility characteristics of our implementation for a suite of sensor and IoT datasets (Section VI). This paper makes the following contributions:

- This is the first work exploring feasibility of supporting local differential privacy on ULP systems. We show that the low resolution and fixed point nature of ULP hardware prevents local differential privacy guarantees from being provided with a conventional noising implementation.
- We propose a modified implementation supporting resampling and thresholding to guarantee local differential privacy on ULP hardware. These techniques generate requisite noise distribution in spite of low resolution, fixed point hardware implementation. To the best of our knowledge, this is the first work presenting local differential privacy implementation on ULP hardware.
- We propose the first privacy budget control algorithm for local differential privacy on ULP hardware. The algorithm reduces the total privacy loss over an aggregate set of queries.
- We present the first evaluation of hardware support for local differential privacy for ULP systems. Our hardware implementation, DP-Box, has low overhead and provides high utility, while guaranteeing local differential privacy, over a range of sensor/IoT benchmarks. We show that DP-Box can also be easily configured to support Randomized Response [19] for categorical data privacy.

The rest of the paper is organized as follows. In Section II, we provide background on differential privacy and how it works in a sensor and IoT environment. Section III points out the vulnerability of differential privacy implemented on fixed-point hardware and analyzes in detail how to implement and guarantee differential privacy for ULP systems. Section IV discusses in detail our proposed DP-box architecture. Section V describes how we evaluate the performance of the DP-box. Section VI presents experimental results. Section VII concludes.

II. Background

A. Differential Privacy

Assume that there exist two identical numeric databases D_1 and D_2 except one element: D_1 with a person A's information and D_2 without it. Such databases D_1 and D_2 are called *adjacent*. In that case, we get two different query outputs when applying the same statistical query like mean on those two databases. Then, the person A's information can be revealed from the two query outputs, which implies that returning exact query output can leak private information entirely.

Differential privacy (DP) is a standard notion of privacy that was originally developed by Dwork, McSherry, Nissim, and Smith [20] and has gained significant attention over the years [14]. DP ensures that the probability that a statistical query will output a given result is almost the same whether it is applied on D_1 or D_2. In more detail, a randomized mechanism $\mathscr{A}$ gives ε-DP if for all adjacent databases D_1 and D_2 and all $S \subset Range(\mathscr{A})$,

$$\Pr[\mathscr{A}(D_1) \in S] \leq \exp(\varepsilon)\Pr[\mathscr{A}(D_2) \in S], \quad (1)$$

where $\mathscr{A}(D_1)$ is the random output of the mechanism $\mathscr{A}$ when the query f is applied to the database D_1. In other words, given a DP output, the likelihood ratio between any adjacent two databases is similar as shown in Fig. 1 and furthermore bounded by $\exp(\varepsilon) \approx 1+\varepsilon$ when ε is small. Thus smaller ε provides higher level of privacy.

While there are several mechanisms to implement differential privacy [21], the most popular technique is the Laplace mechanism [20], which adds random noise with a zero-mean Laplace distribution to the query output $f(D)$. The standard deviation of the Laplace distribution is scaled depending on ε and the *(global) sensitivity* of a query f. The sensitivity $GS(f)$ of a query f is defined as

$$GS(f) = \max_{\text{adjacent} D_1, D_2} |f(D_1) - f(D_2)|. \quad (2)$$

For instance, the sensitivity of the counting query is 1 because the maximum counting difference between any two adjacent databases is 1. Similarly, it is possible to calculate the sensitivity of other statistical queries like mean, variance, and median. The Laplace mechanism generates output, as shown in Fig. 1, as a summation of the original query output $f(D)$ and random noise sample from $\text{Lap}(GS(f)/\varepsilon)$, i.e. $\mathscr{A}(D) = f(D) + \text{Lap}(GS(f)/\varepsilon)$, where $\text{Lap}(\lambda)$ is a random variable with probability density function

$$f(x) = \frac{1}{2\lambda}\exp(-\frac{|x|}{\lambda}), \qquad \forall x \in \mathbb{R}. \quad (3)$$

It can be easily proved that the Laplace mechanism satisfies (1) and guarantees ε-DP. One thing to note is that noise scaling for DP does not require knowing the contents in the database but it is dependent only on the sensitivity of the query ($GS(f)$) to be computed and the acceptable amount (ε) of privacy.

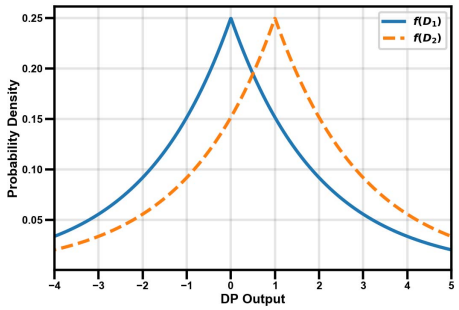

Figure 1: Differential privacy.

In order to quantify the privacy leak from answering a query, we refer to the quantity

$$\ell^{(y)}_{D_1,D_2} = \log\frac{\Pr[\mathscr{A}(D_1) = y]}{\Pr[\mathscr{A}(D_2) = y]} \quad (4)$$

as the *privacy loss* incurred by reporting the value y for adjacent databases D_1 and D_2. Intuitively, infinite privacy loss implies that the private information will be disclosed entirely

when reporting DP output as y, and small privacy loss means little information leakage. It can be easily seen that the Laplace mechanism for ε-DP theoretically guarantees that the privacy loss is always less than ε for all the possible DP output. This is important to note because in later sections we show that in practice the privacy loss can not be bounded even if the Laplace mechanism is implemented in ULP systems.

Rather than simply reporting the exact query output, DP adds error by perturbing the output; thus masking any individual private information. However, while the amount of privacy loss from a single query can be bounded by a small value, this loss starts accumulating as someone makes more queries on the database. The composition theorem [22] states how much total privacy loss in the worst case is incurred by making multiple queries on a given database: when a series of queries $(f_1, f_2, ..., f_n)$ is applied and each gets output with ε_i-DP, the overall privacy loss is $\sum_{i=1}^{n} \varepsilon_i$. This implies that implementing a DP mechanism alone is not enough and we also need a means to limit the total privacy loss. The total allowed loss is referred to as *privacy budget*, which determines how many queries will be allowed to ask.

A final remark on DP is that there exists a fundamental tradeoff between privacy and utility, or accuracy, with the usage of DP. Utility is often measured with mean absolute error (MAE) of the DP output with respect to the accurate query output. For instance, if ε is set too high, we get more accurate output with higher probability but sensitive data will leak. On the other hand, small ε will provide better privacy, but the DP output might not be particularly useful due to large error.

B. Local Differential Privacy

Conventional DP in most of the literature has assumed that there exists a trusted database where all the original data are collected and that queries are executed by the trusted database operator, or data curator, who can have access to all of the raw data. This model requires the data curator to collect all of the raw sensor data information into a massive centralized database and then calculate privacy-preserving statistics on it, and the user cannot help trusting the data curator. This would make the private data vulnerable to many potential attacks.

Theoretically it is also possible to apply DP in the *local setting*, where there is no trusted data curator, and each sensor submits the private data after adding noise individually [23]. The comparison between the conventional DP and local DP is depicted in Fig. 2. This approach is quite simple to implement and promising in privacy perspective since it does not collect the raw data at all. In essence, this is a generalization of Randomized Response [19] applied to real-valued numeric data.

In the local DP setting, each user does not want their private data to be revealed by others who have access to the sensor data. In other words, given a noised output from a sensor, it should be unable to tell the original data because all the possible sensor data have similar probabilities to report the given noised output. Mathematically, this can be stated as: in order to guarantee ε-DP for each sensor data, it is required that for arbitrary two possible inputs x_1 and

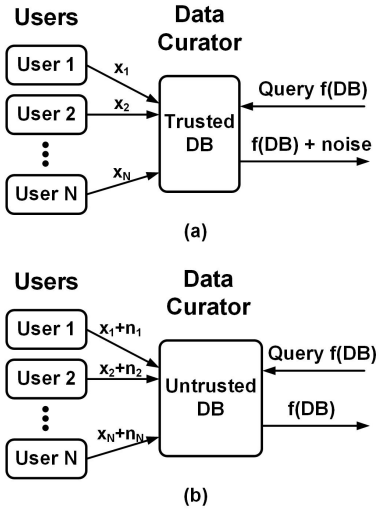

Figure 2: Differential privacy in different settings: (a) conventional and (b) local setting.

x_2, and any noised DP output y,

$$\Pr[y|x_1] \leq \exp(\varepsilon)\Pr[y|x_2] \tag{5}$$

One way to satisfy the condition (5) is to apply the Laplace mechanism locally. If the sensor data x is within the range $[m,M]$ with the length $d=M-m$, then it can be proved that the Laplace mechanism $y=x+n$ with $n \sim \mathrm{Lap}(d/\varepsilon)$ satisfies (5). This is a very strong privacy guarantee because, even if an adversary uses his side information and makes the estimation of the original sensor data into a binary choice problem, the likelihood ratio between two hypotheses is bounded by $1 \pm \varepsilon$ when ε is low. Since they are almost equally likely, it is possible to keep the data private from the adversary.

Statistical queries such as mean, median, or standard deviation, can be applied to noised data as shown in Fig. 2(b) to obtain aggregate information of raw data while preserving individual privacy. In fact, applying any function or query on DP output still preserves privacy due to the following. Assuming that the outputs of other users are known, the query output $f(x)$ is a function of a sensor output x_1 that the adversary wants to disclose. Then, by (5) (denoting g as the local DP mechanism in the sensor, i.e. DP output $y=g(x)$)

$$\begin{aligned}\Pr[f(g(x_1))=t] &= \sum_{y \in \{s|f(s)=t\}} \Pr[f(y)=t]\Pr[g(x_1)=y] \\ &\leq \sum_{y \in \{s|f(s)=t\}} \Pr[f(y)=t]\exp(\varepsilon)\Pr[g(x_2)=y] \\ &= \exp(\varepsilon)\Pr[f(g(x_2))=t],\end{aligned} \tag{6}$$

which implies that different sensor data can generate the same query output with similar probability, thereby satisfying ε-DP. Another benefit of applying local DP in each sensor is that it provides a principled way to add noise to the raw sensor data in a way that allows the system to measure exactly how much privacy is lost in responding to each sensor data request. In the following sections, we explain how to implement a hardware that supports local DP for ULP systems and evaluate the

utility of the statistical queries applied on noised data.

III. Implementing Local Differential Privacy on ULP Hardware

Differential privacy has largely been used in context of large-scale databases, where energy consumption is not a constraint. The critical issue in implementing local differential privacy for ULP systems is that energy consumption becomes a crucial constraint. As one example, ULP systems support fixed point hardware, not floating point hardware, for cost, area, energy, and latency reasons. No prior work considers the feasibility of implementing differential privacy in such systems.

A. Local Differential Privacy on Fixed-Point Hardware

Although fixed-point hardware may seem suitable to implement local differential privacy on ULP systems, its limited number of bits to represent numbers can lead to undesired privacy loss, as described below.

1) Random Number Generation: One of the major components to implement differential privacy is a random number generator (RNG). As explained in the previous section, to guarantee differential privacy, noise that is added to the real data needs to have a certain probability distribution, and the most widely used one is the Laplace distribution [20]. While there are several techniques to generate random numbers [24], the inversion method [25], [26] is the most commonly used in energy constrained settings. In order to use the inversion method, the cumulative distribution function (CDF) of the desired random variable must first be calculated. Let $F_X(x)$ denote the CDF of a random variable X. Then the inversion method takes the output u of a uniform random number generator (URNG) between 0 and 1 as an input and maps it to the output x, the inverse CDF (ICDF) output of u, i.e. $x = F_X^{-1}(u)$. This method is especially well suited for generating the Laplace distribution because 1) it is symmetric around the mean and 2) it has the closed-form expressions for the CDF and ICDF, which makes it easy to analyze. Due to symmetry, half of the ICDF $F_X^{-1}(u)$ of a zero-mean Laplace distribution with parameter λ, Lap(λ), can be represented as

$$F_X^{-1}(u) = -\lambda \log(u) \quad (0 < u \leq 1), \tag{7}$$

where log is logarithm with base e. Thus, noise n from the Laplace distribution can be generated from two independent random numbers u_1,u_2 from URNG.

$$n = \lambda \,\mathrm{sgn}(u_1 - 0.5)\log(u_2) \tag{8}$$

In (8), sgn(x) determines the sign of the random number. As shown in (8), generating every single noise sample requires a potentially expensive logarithm operation. However, approximations of (8) can be made using CORDIC algorithm or a number of polynomial segments of low degree, which has been applied in many energy-efficient fixed-point RNG hardware works [25], [26]. In the following, we analyze the impact of fixed-point representation on noise distribution and further its impact on local differential privacy.

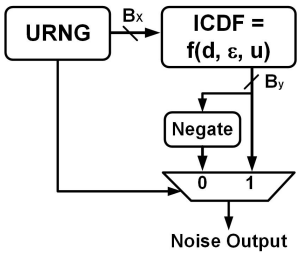

Figure 3: Simplified block diagram of fixed-point RNG hardware.

2) Noise Distribution of Fixed-Point RNG Hardware: Figure 3 shows the simplified block diagram of the fixed-point RNG hardware using inversion method. Let B_x and B_y denote the number of output bits of a URNG and RNG hardware, respectively. Let us assume that the sensor output x lies in the range of $[m,M]$ ($m \leq x \leq M$) with the length $d = M - m$, i.e. the data measured with a sensor is a value between m and M. Then theoretically it is possible to achieve ε-DP using the simple additive noise mechanism with noise distribution Lap(d/ε) as explained in the previous section. In other words, by adding $n \sim$ Lap(d/ε) to the sensor output x, reporting $x+n$ enables the specific sensor output to hide among many other sensor outputs.

Let us first see how the RNG output distribution looks like with fixed-point representation. The URNG output u with B_x bits and the RNG output n with B_y bits can be represented as

$$u = m2^{-B_x} \quad (m \in \{1,2,...,2^{B_x}\}) \tag{9}$$

$$n = k\Delta \quad (k \in \{-2^{B_y-1},...,-1,0,1,...,2^{B_y-1}-1\}), \tag{10}$$

respectively; here Δ is the quantization step decided by the resolution. As shown in Fig. 3, in fixed-point RNG hardware, each URNG output u in (9) is mapped to a value $-\frac{d}{\varepsilon}\log(u)$ by (7), and then it is rounded to a nearest value $k\Delta$ in (10), and finally its sign is determined. Figure 4 illustrates the comparison between the ideal Laplace distribution (Lap(20)) and the fixed-point (FxP) Laplace RNG distribution with $B_x = 17$, $B_y = 12$, and $\Delta = 10/2^6$. As shown in Fig. 4(a), the FxP RNG output shows almost the identical result to the ideal Laplace distribution in the area where the density is high. However, if we zoom into the region near the tail (see Fig. 4(b)), we can clearly see the difference between the ideal distribution and the FxP RNG distribution. Essentially, the FxP RNG output distribution shows the discrete probability values, which are multiples of $\frac{1}{2^{B_x+1}}$, while the ideal distribution has continuous density. Also, the output generated by the FxP RNG is bounded, while the ideal distribution has an unbounded range. In fact, from (7) and (9), we can see that the maximum value that the FxP RNG can generate is $\frac{d}{\varepsilon}B_x\log 2$. Since B_x plays a critical role in these two nonidealities, we can increase B_x and make the FxP RNG output distribution more close to the ideal one. However, as long as B_x is finite, which is the case in any FxP RNG hardware, there always exists a large difference in the tail region. In the following, we will see how this nonideality in distribution affects the privacy loss defined in (4).

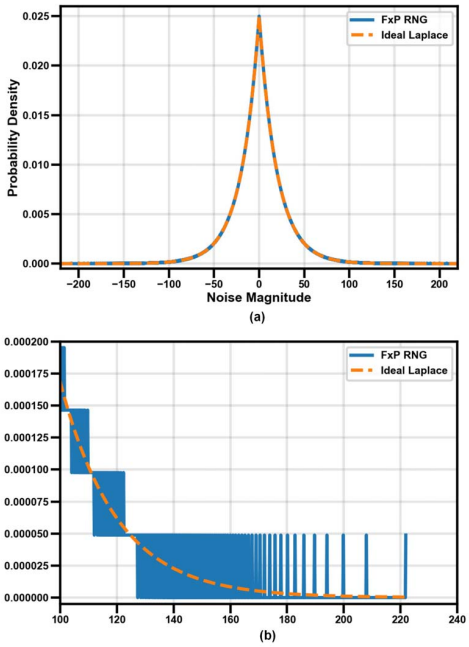

Figure 4: Comparison between the ideal Laplace distribution and fixed-point RNG output: (a) overall distribution and (b) zoomed-in tail distribution.

3) Impact on Differential Privacy: The main result of this section is that a naive implementation of the conventional DP mechanism in FxP hardware cannot guarantee DP. In other words, implementing DP mechanism without considering the FxP RNG architecture may disclose the entire information about the sensor data even if the system reports the noised data.

To see why, let us first consider the ideal case. Recall that, in order to guarantee ε-DP, the noise $n \sim \text{Lap}(d/\varepsilon)$ needs to be generated to implement the Laplace mechanism. In the ideal case, since $n \in (-\infty,\infty)$, applying the Laplace mechanism to sensor output $x \in [m,M]$ generates any value in $(-\infty,\infty)$ with positive probability. Thus, for any value of the noised output, the privacy loss is bounded by ε.

However, as shown in the previous section, the FxP RNG output range is always bounded. In other words, the noise n generated by the FxP RNG is limited within a range $[-L,+L]$, instead of $(-\infty,\infty)$. Thus, when the conventional Laplace mechanism is used, each sensor output will yield a different range of the noised output. For example, let us consider two cases: when the sensor output is either m or M. When the Laplace mechanism applied with the FxP RNG, if the data is m, the noised output will be some value in the range $[-L+m,L+m]$. On the other hand, if the data is M, the noised output has a different range $[-L+M,L+M]$ as shown in Fig. 5. In this case, if the reported noised output is lower than $-L+M$, it is possible to know that the original sensor data cannot be M with probability 1. Similarly, if the noised output is larger than $L+m$, the original data cannot be m. Furthermore, if the noised output is either $-L+m$ or $L+M$, then we can exactly know that the sensor data is m or M, respectively. In terms of privacy loss, this is the case when privacy loss becomes infinite because there exist some sensor data that cannot generate the reported noised output. Since differential privacy is guaranteed only when privacy loss is bounded over the entire range of the noised output (see Section II-A), **we conclude that differential privacy is not guaranteed when the Laplace mechanism is implemented naively with FxP hardware**.

Moreover, looking at Fig. 4(b), we can find out that the FxP RNG cannot generate all the noise values that have zero probability in the range $[-L,+L]$. This is because FxP RNG cannot generate a number with arbitrarily small probability due to the limited resolution. As shown in Fig. 4(b), some noise values whose probability in ideal distribution is lower than $\frac{1}{2^{B_x+1}}$ cannot be generated. When noise with this distribution is added to the sensor data, privacy loss again becomes infinite for some noised output range. Therefore, simply implementing FxP RNG for the Laplace distribution and applying the Laplace mechanism in FxP hardware cannot guarantee local DP.

4) Problem Generalization: It may appear that the infinite privacy loss problem is caused by FxP hardware implementation, but, in essence, it originates from the fact that the numbers representable in digital computers are quantized with finite precision (even if we use ultra long floating point numbers). The limited number of bits to represent real numbers fundamentally poses two problems to implementing ideal random number distribution guaranteeing DP, such as Laplace, Gaussian [20], or staircase [21] distributions, either in hardware or in software:

1) Ideally, a DP-guaranteeing RNG should generate any number from $-\infty$ to $+\infty$ with positive probability (e.g. Laplace or Gaussian). However, due to the limited number of bits, there exist minimum and maximum values that any digital hardware can represent.

2) In addition, the random numbers are generated from a uniform random number generator output by applying some transformation, and due to this fact, we can only assign a quantized probability mass to each random number output. However, all the distributions that guarantee DP (e.g. Laplace, Gaussian, or staircase) have arbitrarily small probability at their tail regions, which cannot be represented with finite precision.

Due to these two reasons, any random number (continuous random variable) generated by digital systems cannot have the ideal distribution and will, therefore, not guarantee DP. In fact, [27] points out that naive software implementation of a DP mechanism using floating point numbers also suffers from infinite privacy loss for the same reason. In general, any systems with finite precision to represent real numbers, which is always the case since infinite number of bits cannot be implemented in digital hardware, will have similar problems [28].

5) Comparison against prior work: While [27] focuses on how floating-point arithmetic can impact differential privacy, our work focuses on fixed-point hardware. [28] notices that precision limitations can cause privacy leak,

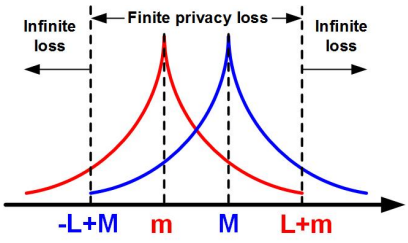

Figure 5: Implementing Laplace mechanism with FxP hardware.

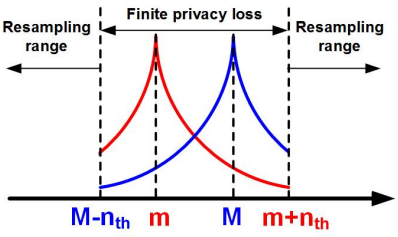

Figure 6: Laplace mechanism with resampling.

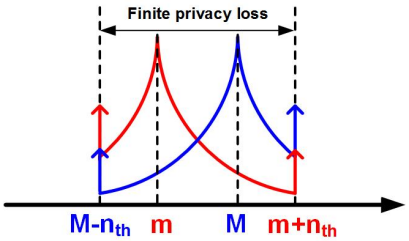

Figure 7: Laplace mechanism with thresholding.

but does not focus on implementation. Its analysis cannot be applied in practice because unless the real distribution of the implemented RNG is known in advance, one cannot calculate its impact on privacy.

Unlike [27], [28], our paper describes the details and analytical expression of impact on data privacy caused by hardware design parameters such as sensor resolution and the number of bits used in fixed-point hardware (Section III-A). Also, we propose two simple privacy-guaranteeing operations, which can be built in hardware with minimal overhead (Section III-B). We also evaluate the utility for some aggregate statistical queries like mean, variance, etc. using the real datasets with the proposed DP-box (Section VI). Finally, we show that we can perform both effective budget control and support other privacy-preserving techniques such as Randomized Response (Section VI).

B. Enabling Local Differential Privacy Guarantees

Here, we propose two new ways to 1) enable local privacy guarantees on FxP hardware and 2) bound the privacy loss to the level we want.

It can be proven that the FxP RNG described in Section III-A2 generates the random number n with the following probability.

$$\begin{aligned} \Pr[n=k\Delta] &= \frac{\lfloor m_1(k)\rfloor - \lceil m_2(k)\rceil + 1}{2^{B_x+1}} \\ m_1(k) &= \exp(B_x\log 2 - \varepsilon\frac{\Delta}{d}(k-\frac{1}{2})) \\ m_2(k) &= \exp(B_x\log 2 - \varepsilon\frac{\Delta}{d}(k+\frac{1}{2})) \end{aligned} \tag{11}$$

where $\lfloor x\rfloor$ and $\lceil x\rceil$ represent the largest integer less than or equal to x and the smallest integer greater than or equal to x, respectively.Since the privacy loss is a function of the FxP RNG output, or k, and the loss tends to increase as the noise magnitude increases (k is large), it is natural to consider limiting the magnitude of the noised output in order to bound the privacy loss and guarantee local DP.

For limiting the noised output magnitude, we propose two independent mechanisms, resampling and thresholding, which will be explained in detail next.

1) Resampling: What resampling does is that when the noised output magnitude is larger than a preset threshold value, the FxP RNG resamples the noise until the noised output is within the required range. Without resampling, the overall noised output range will be $[-L+m,L+M]$ due to $n\in[-L,+L]$ and $x\in[m,M]$. When resampling is used, the noised output range is limited to $[-n_{th1}+m,n_{th1}+M]$, where n_{th1} is the threshold for resampling. As shown in Fig. 6, the noised output range of all the sensor data x has the same range in this case, thus the privacy loss is bounded. If we set n_{th1} small, then the privacy loss will be close to the ideal case, but the noise needs to be resampled more frequently, which degrades the energy efficiency. On the other hand, if n_{th1} is set to be large, then we can achieve good energy efficiency, but the maximum privacy loss is increased too much. Thus, it is important to set the threshold n_{th1} properly.

Thankfully, the threshold n_{th1} required to bound the privacy loss to the desired level can be directly calculated as a function of d, Δ, ε, and B_x. For example, if we want to bound the maximum privacy loss to be $n\varepsilon$ when $\mathrm{Lap}(d/\varepsilon)$ is implemented in FxP RNG, from (11) it should satisfy:

$$\frac{\lfloor m_1(k)\rfloor - \lceil m_2(k)\rceil + 1}{\lfloor m_1(k+\frac{d}{\Delta})\rfloor - \lceil m_2(k+\frac{d}{\Delta})\rceil + 1} \le \exp(n\varepsilon) \tag{12}$$

Using the fact that $m_1(k) - 1 < \lfloor m_1(k)\rfloor \le m_1(k)$ and $m_2(k) \le \lceil m_2(k)\rceil < m_2(k)+1$, (12) yields

$$k \le -\frac{1}{2} + \frac{d}{\varepsilon\Delta}(B_x\log 2 + \log\frac{(\exp(\varepsilon\frac{\Delta}{d})-1)(\exp((n-1)\varepsilon)-1)}{1+\exp(n\varepsilon)}). \tag{13}$$

In other words, setting the threshold as $n_{th1} = d - \frac{\Delta}{2} + \frac{d}{\varepsilon}(B_x\log 2 + \log\frac{(\exp(\varepsilon\frac{\Delta}{d})-1)(\exp((n-1)\varepsilon)-1)}{1+\exp(n\varepsilon)})$ and implementing the Laplace mechanism with resampling ensures the privacy loss less than $n\varepsilon$ and also guarantees local DP.

2) Thresholding: Thresholding is similar to resampling, but instead of resampling the noise when the noised output magnitude is larger than a preset threshold value, the noised output is rounded to the threshold. When the threshold is set to be n_{th2}, the overall noised output range becomes $[-n_{th2}+m, n_{th2}+M]$ as shown in Fig. 7. Since all the noised outputs outside the region $[-n_{th2}+m, n_{th2}+M]$ are mapped to either $-n_{th2}+m$ or $n_{th2}+M$, we can see higher probabilities at these minimum and maximum values. Because thresholding generates different output distribution from resampling, the threshold value to bound the privacy loss and to guarantee local DP will be different.

Again, the threshold can be calculated as a function of d, Δ, ε, and B_x. As before, let us assume that we want to bound the maximum privacy loss to be $n\varepsilon$.Since $\Pr[n\ge k\Delta] = \frac{\lfloor m_1(k)\rfloor}{2^{B_x}}$, the condition that the privacy loss at

the boundaries should be less than $n\varepsilon$ can be represented as:

$$\frac{\lfloor m_1(k)\rfloor}{\lfloor m_1(k+\frac{d}{\Delta})\rfloor} \leq \exp(n\varepsilon) \tag{14}$$

This yields

$$k \leq \frac{1}{2} + \frac{d}{\varepsilon\Delta}(B_x \log 2 + \log(\exp(-\varepsilon) - \exp(-n\varepsilon))) \tag{15}$$

Thus, setting the threshold as $n_{th2} = d + \frac{\Delta}{2} + \frac{d}{\varepsilon}(B_x \log 2 + \log(\exp(-\varepsilon) - \exp(-n\varepsilon)))$ and implementing the Laplace mechanism with thresholding ensures the privacy loss less than $n\varepsilon$ and also guarantees local DP.

3) Why Thresholding/Resampling Preserve Privacy: Intuitively, data is kept private in local DP by ensuring that all data have roughly equal likelihood of producing a given noisy output (Fig. 1). As long as this condition is guaranteed, the actual shape of the distribution does not matter. Indeed, thresholding changes the shape of the distribution (see Fig. 7), but an intelligent choice of threshold (n_{th}) guarantees that original data cannot be inferred for a given noisy output (specifically, n_{th} is chosen such that worst-case log likelihood ratio is bounded by $n \times \varepsilon$). In Fig. 7, for example, n_{th} is chosen so that both data m and M have similar probability to report the boundary values. If an adversary receives the DP output at boundary values, since the probabilities that the original data is m or M are similar, the adversary cannot estimate the original data.

Both resampling and thresholding guarantee local DP implemented in FxP hardware, but they have different energy efficiency and add different amount of noise to the sensor data. With resampling, FxP RNG needs to operate multiple times when noised output is outside the required range, which increases the average energy consumption. On the other hand, with thresholding, one noise sample is enough to generate the noised output, thus having better energy efficiency. However, as shown in Fig. 6 and Fig. 7, the probability distributions of the noised output between two techniques are different, which results in different utility when some query is applied. Comparison between two methods will be explained more in Section VI with the experimental results.

C. Budget Control on Fixed-Point Hardware

Budget control is needed to guarantee privacy over multiple queries as well as to prevent a malicious, or negligent, application or user from causing an unbounded loss in privacy.To the best of our knowledge, no previous work has proposed a budget control algorithm for local DP, let alone local DP on ULP hardware. One simple way to implement budget control for local DP is by simply counting the number of requests and limiting the maximum number of requests to some value. However, as explained in the previous section, since the privacy loss when DP is implemented in FxP hardware is different from the ideal case, the privacy budget control logic needs to be modified accordingly.

We propose a privacy budget control algorithm that calculates privacy loss dependent on the noised output value in FxP hardware implementation. Using the analysis that we have done in the previous section, it is possible to divide the overall noised output range into segments with different privacy loss.

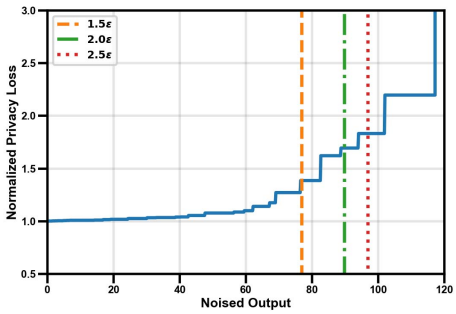

Figure 8: Privacy loss as a function of noised output.

For example, take a look at Fig. 8, which shows the normalized privacy loss as a function of noised output values. Since the output distribution is symmetric, we only show the loss in the region where the output values are larger than M in Fig. 8. The dashed lines represent the threshold values corresponding to each privacy loss. As an example, if the noised output is within the range $(M, M+76]$ and $(M+76, M+90]$, then the privacy loss is no more than 1.5ε and 2.0ε, respectively. Thus, depending on the noised output value, our privacy budget control algorithm adjusts the privacy loss for each request adaptively. Algorithm 1 shows an example privacy budget control algorithm for the case when the noised output range is divided into seven segments and thresholding is implemented. If resampling is used, instead of $\ell = \varepsilon_2$ in the last else statement, we should resample a new noise, update *tmp*, and check again at which segment the new *tmp* falls in.

In Algorithm 1, the control logic stops providing outputs to the sensor data requests after all the privacy budget is consumed. One practical way to give outputs to data requests without more privacy loss after the budget is used up is caching, which simply returns the cached result without calculating a new noised output. Since caching keeps using the old output repeatedly and does not provide any additional information, there is no more privacy loss while responding to all the data requests.

D. Hardware versus Software Support for Privacy

Implementing DP mechanism and privacy budget control in software would allow support on existing microcontrollers without modification. In spite of this, we propose custom hardware support for the latency, energy, and security benefits. We found that to support sensors with resolution up to 13 bits with privacy parameter $\varepsilon \geq 0.1$, we needed to use 20-bit fixed-point values. The latency of generating a 20-bit fixed-point sample from the Laplace distribution and noising the sensor value in software is 4043 cycles. The latency of the corresponding software using half precision floating point values is 1436 cycles. These latencies are without any budget update computation. Our hardware implementation noises a sensor value and updates the budget in a single cycle, although we conservatively assumed a latency of four cycles to account for one memory write instruction and one memory read instruction on the MSP430. As a result of these

Input: private sensor data x, noise
n, and a stream of sensor data requests $\{r_i\}$
Output: noised output y
Initialize the privacy budget B;
Let $\ell = 0$;
for *each data request* r_i **do**
  if B ¿ 0 **then**
    Let $tmp = x+n$;
    if $tmp \in [m,M]$ **then**
      $\ell = \varepsilon_{RNG}$;
    else if $tmp \in [m-n_1,m) \cup (M,M+n_1]$ **then**
      $\ell = \varepsilon_1$;
    else
      if $tmp \in [m-n_2,m-n_1) \cup (M+n_1,M+n_2]$ **then**
        $\ell = \varepsilon_2$;
      else
        $\ell = \varepsilon_2$;
      end
    $y = m-n_2$ if $tmp < m-n_2$;
    $y = M+n_2$ if $tmp > M+n_2$;
    $y = tmp$ otherwise;
    $B = B-\ell$;
  else
    Halt.
  end
end

Algorithm 1: Proposed privacy budget control algorithm with thresholoding.

latency savings, we see energy benefits of 894× and 318× compared to the fixed-point and floating-point software implementations, respectively. These energy benefits also do not account for our ability to avoid waking up the microcontroller on every sensor output to perform noising.

In addition to the quantitative benefits, there are also some qualitative security benefits. Microcontrollers like TI's MSP430 lack any memory protection unit, making it impossible to protect vulnerable regions of memory from malicious software. Therefore, implementing privacy in custom hardware is the only way to guarantee that it is not tampered with. Even on systems with some form of memory protection, a hardware implementation guarantees that even completely trusted code cannot manipulate privacy levels.

IV. DP-Box: Hardware Support for Local Differential Privacy

The above mechanisms (noising, resampling, thresholding, and budget control) must be implemented such that untrusted software only has access to noised outputs. This can be accomplished in different ways depending on what components of the system are trusted. In microcontrollers without any mechanisms for process isolation, no software can be trusted. This means that privacy parameters (e.g., the ε value and sensor range d to use for noising, as well as the budget B and budget replenishment period) must be implemented on hardware. If there is more than one sensor, there also may need to be a hardware mechanism for sharing the budget between all sensors since the readings of different sensors could be combined to compromise privacy [29]. If some software can be trusted (e.g., a trusted OS or sensor co-processor), then we can move the aforementioned tasks into software. However, a hardware implementation may still be needed since software may have unacceptably high overhead (Section III-D)

For the remainder of the paper, we will consider the case where some software is trusted. This means that only the noising needs to be implemented in hardware because of the unacceptable latency and energy consumed by the software noising implementation.We present *DP-Box*, a hardware module supporting local differential privacy which could be implemented in either the main processor or the sensor controller (Fig. 10). The presented hardware is not specific to any type of sensor and requires no knowledge of the sensors, except for the sensor range. DP-Box shares some similarities in it's high-level structure with the Privacy Preserving Unit (PPU) [30]. However, unlike the PPU work, we present a specific implementation and evaluate it in different contexts (Section V).

Although budget control can be performed in software, considering the earlier-mentioned strong reasons we had for implementing noising/thresholding/sampling in hardware, we simply embedded the budget control logic into DP-Box at only 11% additional hardware overhead. We use caching and budget replenishment to deal with budget overruns.

A. Hardware Interface

In the current implementation, the main processor can send to the DP-Box the following commands using a 3-bit command port:

- **Start Noising** - Used to indicate that a sensor value x, privacy parameter ε, and sensor range d have been loaded and noising should proceed. When in the initialization phase, it is used to indicate that the budget and replenishment period have been configured and a transition to the waiting phase should be triggered.
- **Set Epsilon** - Used to set the privacy level, ε_{in}, for the next sensor reading. Refer to (19) to understand how ε_{in} relates to the ε used for noising. When in the initialization phase, used to set the budget.
- **Set Sensor Value** - Used to set the sensor value; the value that will be noised.
- **Set Sensor Range, Upper** - Used to set the upper range of the sensor, which will be used for thresholding and re-sampling. When in the initialization phase, used to set the replenishment period.
- **Set Sensor Range, Lower** - Used to set the lower range of the sensor, which will be used for thresholding and re-sampling.
- **Set Threshold** - Used to toggle the DP-Box between re-sampling and thresholding; needs to be re-sent to toggle again.
- **Do Nothing** - This command is used to hold the DP-Box in the idle state, if not used, the DP-Box would immediately begin noising the sensor value again. This command is

included because the sensor value, the sensor range, and the privacy level do not have to change between noising.

A second port is added for the input value. This port takes a signed value as input, and is used by every command above which involves setting a parameter. There are two output ports, a signed output port for the noised value, and a single bit output used to indicate that noising is complete. The ready indicator is necessary because the amount of time each noising will take, when using resampling, is non-deterministic. Finally, the DP-Box also takes a clock as input.

B. Generating Noise

A Laplace RNG using inverse CDF is implemented in hardware, using (17), and samples are computed when needed. Then the Laplace sample is scaled by s_f, which is defined by (16). Notice that s_f depends on the sensor range, and the privacy parameter ε; generated noise cannot be used on a sensor value until all three parameters have been set. The sensor range is calculated based on the upper and lower limits, r_u and r_l respectively. However, the sample from the Laplace distribution can be calculated immediately, and does not need to immediately change as s_f changes. (17) depends only on a uniform random number u, where $0 < u \leq 1$.

$$s_f = \left(\frac{r_u - r_l}{\varepsilon} \right) \tag{16}$$

$$l_s = \begin{cases} \log(2 \times u) & u < 0.5 \\ -\log(2 \times (1-u)) & u \geq 0.5 \end{cases} \tag{17}$$

$$n = s_f \times l_s \tag{18}$$

$$\varepsilon = 2^{-\varepsilon_{in}} \tag{19}$$

By implementing a CORDIC logarithm function and paying a higher area penalty, the entire logarithm computation can be completed in a single cycle. The uniform random number u used in (17) comes from a Tausworthe random number generator [25]. By using the privacy parameter ε as (19), as multiplication can be implemented using bit shift, the latency of calculating the scaling factor is also minimized.

C. Operation

The operation of the proposed hardware (see Fig. 9) can be described in three phases:

1) Initialization: When the DP-Box is set up during the system boot process, before the majority of the operating system has been loaded, it is said to be in the intialization phase. While in the initialization phase, the budget and replenishment period are configurable. The integrity of the budget and replenishment period are easier to guarantee in the initialization phase considering features such as secure boot [31] available in today's systems. Budget and replenishment period are not allowed to be changed after the initialization phase. A transition from the initialization phase into the waiting phase occurs when the main processor sends Start Noising command, and the DP-Box cannot return to the initialization phase until the system is power cycled.

2) Waiting for Noise Request: Once initialized, the DP-Box enters a phase where it waits for the next noise request. Although the DP-Box is idle, from the point of view of the main processor, it is not idle internally. While in this phase, the DP-Box keeps track of the time since the last budget replenishment, and resets the budget once the appropriate amount of time has elapsed. The DP-Box also generates a new noise sample immediately upon entering this stage.

Referring to (16), it becomes clear that a new sample can be generated as soon as ε_{in}, r_u, and r_l have been configured; however, we refrain from computing the outcome of (18) until entering the noising phase. As the upper and lower ranges of a sensor are likely to both change when one changes, computing the noise when either changes results in a wasted computation; increasing energy consumption.

Upon entering the waiting phase, however, a new sample from the Laplace distribution is immediately computed (l_s in (17)). This allows for noising to occur in a single cycle within the noising phase. A single cycle, naturally, is the best case; re-sampling can further affect the latency (see Section VI).

3) Noising: Whereas the waiting phase computed the outcome of (17), the noising phase computes the output of (18). This output is then used for re-sampling and thresholding.

If the addition of the generated noise with the sensor value lies outside the acceptable sensor range, and re-sampling is enabled, a new sample from the Laplace distribution must be generated. The latency of the noising mechanism will be equal to the number of times the mechanism re-samples. On every clock cycle, a new sample from the Laplace RNG is generated; the re-sampling logic makes a new comparison every cycle, and immediately noises and returns the sensor value once a valid sample is drawn. By having a new sample ready as soon as the re-sampling logic needs it, the latency is minimized.

Our implementation of resampling may introduce a timing channel since the number of resamples depends on the sensor value, but a straightforward solution to prevent this is to sample noise multiple times instead of only one and choose one of them in the required region.

V. Methodology

We implemented DP-Box in RTL with 20-bit noised output in 65nm technology node. The DP-Box synthesized for an operating frequency of 16 MHz using Synopsys Design Compiler has 10431 gates, critical path length of 58.66 ns, and power of 158.3 uW. The critical path length is adequate for ULP systems since a) they run at relatively low frequency and b) accompanying sensors take 10s of cycles to access [32] (over a serial I2C bus, for example).When no resampling or thresholding is needed, noised output is generated in 2 cycles (one cycle to load the register + one cycle to generate noised output). Thresholding does not require an additional cycle. On resampling, every re-sample requires 1 additional cycle. We generated several other variants of DP-Box to better understand latency / area tradeoffs. Unsurprisingly, we found that pipelined variants reduced critical path length at the expense of area. Similarly, relaxation of timing constraints leads

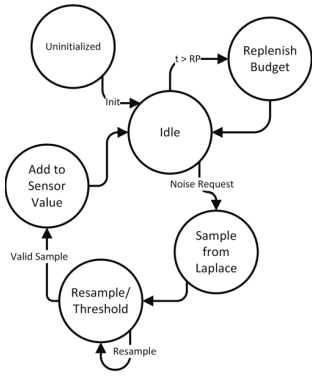

Figure 9: DP-Box operation

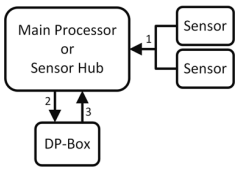

Figure 10: Interface between sensors, the processor, and the DP-Box. The sensor values are first read in and interpreted by the processor, using the sensor drivers (1); in this figure, the sensors share a common bus, e.g. I2C. The interpreted value is then sent to the DP-Box for noising (2), and is read out once the DP-Box sets the 'Ready' line to the processor high (3).

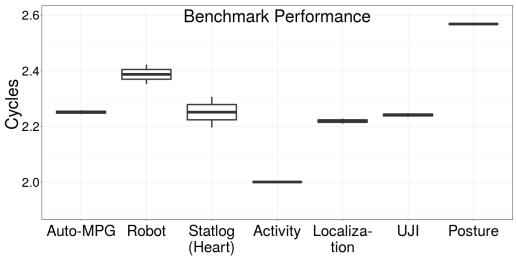

Figure 11: Average latency of the DP-Box, for every sensor type within a dataset.

to area and power reduction (for example, a variant with 30ns critical path length had 9621 gates and consumed 252uW).

In order to both evaluate DP-Box's utility as well as latency (note that latency is data-dependent for resampling), we used seven datasets from the UCI Machine Learning Repository [33], [34], [35], [36], [37], [38], [39]. The chosen datasets (shown in Table I) are diverse and represent a variety of sensor and IoT applications. For each entry in a given dataset, we present it to the DP-Box 500 times to calculate latency and generate the corresponding noised output. Note that a different noise may be generated for each of the 500 samples, so the corresponsing latency and noised output may be different. We report utility of different privacy implementations using the metric *mean absolute error* for mean, median, variance, and counting queries. We report error and latency for a dataset averaged over all the samples in the dataset (number of entries*500).

VI. Results

A. Naive Implementation of Local Differential Privacy on Fixed-Point Hardware

Figure 12 shows the output histogram of the implemented DP-box without resampling or thresholding. Two data from Statlog heart-rate dataset were chosen and fed to the DP-box, and Fig. 12 represents the histogram of all the outputs generated by the DP-box with $\varepsilon = 1$. As shown in Fig. 12(a), the DP-box adds noise with a Laplace distribution to the original data, but in order to verify whether privacy is preserved, we must check if, for any given outputs, both distributions give similar probability. As per our analysis in the previous section, if we zoom in the histogram near the tail region as shown in Fig. 12(b), it can be easily seen that privacy is not preserved because two data can be totally distinguishable if the DP output reports a value that only one data can generate. On the other hand, the proposed DP-box can successfully preserve privacy using resampling or thresholding by limiting the DP output range where both histograms have nonzero counts.

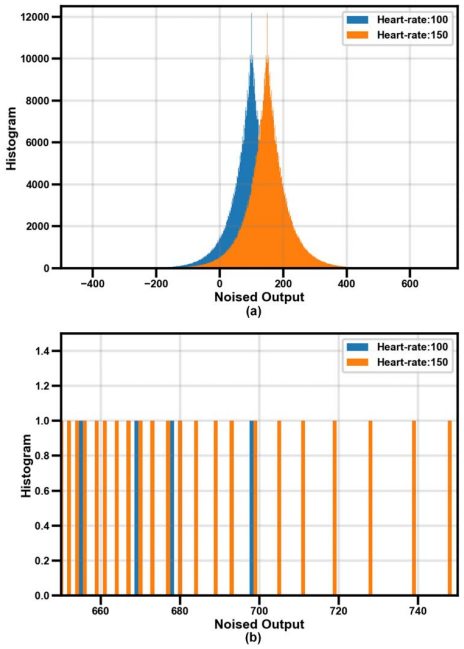

Figure 12: Laplace mechanism output on FxP hardware: (a) overall histogram and (b) zoomed-in tail region.

B. Utility / Latency Analysis

Table II, III, IV, and V show the mean absolute error (MAE) result of the statistical mean, median, variance, and counting queries, respectively. The private data from each dataset were fed into the implemented DP-box, and the mean absolute error as a measure of utility for each query was calculated over noised DP-box output. All of the utility results are for the privacy setting $\varepsilon = 0.5$. We simulated four different settings for comparison. In the first setting ideal Laplace distribution was generated for local differential privacy, and in the second one the Laplace mechanism was implemented on FxP RNG as a baseline. In the third and

Dataset	Entries	Min/Max	Mean	Standard Deviation
Auto-MPG - e.g. model year and MPG [36]	398	9.0/46.6	23.5	7.8
Robot Sensors - e.g. sonar and visual sensors [35].	6129	0.0/360.0	240.4	137.8
Statlog (Heart) - e.g. Blood Pressure [33].	270	94/200	131.3	17.8
Human Activity - Wearable sensors [39].	2650131	-702/533	-36.34	124.04
Localization for Person - Wearable sensors [37].	165633	$-2.54/6.34*10^{17}$	$1.58*10^{17}$	$2.74*10^{17}$
UJIIndoorLoc - WiFi for Localization [38].	19937	-7691.3/-7300.9	-7464.3	123.4
Postural Transitions - Wearable sensors [34].	10929	-10.01/10	0.15	0.52

Table I: Datasets used for utility comparisons.

	Ideal Local DP		FxP HW Baseline		Resampling		Thresholding	
Dataset	MAE	LDP?	MAE	LDP?	MAE	LDP?	MAE	LDP?
Auto-MPG	4.3±3.2 (11%)	Y	4.2±3.2	N	4.1±3.1	Y	4.2±3.1	Y
Robot Sensors	10.3±7.8 (3%)	Y	10.3±7.8	N	9.8±7.5	Y	10.1±7.7	Y
Statlog (Heart)	14.5±11.0 (13%)	Y	14.2±10.9	N	13.8±10.6	Y	14.0±10.7	Y
Human Activity	1.70±1.3 (1.3%)	Y	1.70±1.3	N	1.66±1.3	Y	1.67±1.3	Y
Localization for Person	0.068±0.05 (1.3%)	Y	0.068±0.05	N	0.12±0.07	Y	0.067±0.05	Y
UJIIndoorLoc	6.2±4.7 (1.6%)	Y	6.2±4.7	N	6.0±4.6	Y	6.1±4.6	Y
Postural Transitions	0.11±0.09 (1.5%)	Y	0.12±0.09	N	0.18±0.12	Y	0.11±0.09	Y

Table II: Mean absolute error for mean query.

	Ideal Local DP		FxP HW Baseline		Resampling		Thresholding	
Dataset	MAE	LDP?	MAE	LDP?	MAE	LDP?	MAE	LDP?
Auto-MPG	3.2±2.5 (8.6%)	Y	3.3±2.5	N	3.2±2.4	Y	3.3±2.5	Y
Robot Sensors	87±11 (24%)	Y	87±11	N	88±11	Y	87±11	Y
Statlog (Heart)	11±8.4 (10%)	Y	11±8.5	N	11±8.5	Y	11±8.5	Y
Human Activity	7.3±1.6 (5.1%)	Y	7.3±1.7	N	7.1±1.7	Y	7.3±1.7	Y
Localization for Person	0.061±0.05 (1.2%)	Y	0.060±0.04	N	0.74±0.0	Y	0.060±0.04	Y
UJIIndoorLoc	39±6.3 (10%)	Y	39±6.3	N	40±6.4	Y	39±6.3	Y
Postural Transitions	0.083±0.06 (1.1%)	Y	0.085±0.06	N	0.68±0.4	Y	0.085±0.06	Y

Table III: Mean absolute error for median query.

	Ideal Local DP		FxP HW Baseline		Resampling		Thresholding	
Dataset	MAE	LDP?	MAE	LDP?	MAE	LDP?	MAE	LDP?
Auto-MPG	1010±770	Y	1020±770	N	1020±760	Y	1020±770	Y
Robot Sensors	2.4E4±1.8E4	Y	2.4E4±1.8E4	N	2.4E4±1.8E4	Y	2.4E4±1.8E4	Y
Statlog (Heart)	2.0E3±1.5E3	Y	2.0E3±1.5E3	N	1.9E3±1.4E3	Y	1.9E3±1.5E3	Y
Human Activity	1600±1200	Y	1600±1200	N	1600±1200	Y	1600±1200	Y
Localization for Person	2.2±1.7	Y	2.2±1.6	N	10.1±2.6	Y	2.2±1.6	Y
UJIIndoorLoc	1.5E4±1.2E4	Y	1.6E4±1.2E4	N	1.6E4±1.2E4	Y	1.6E4±1.2E4	Y
Postural Transitions	5.7±4.3	Y	5.7±4.3	N	15.9±6.8	Y	5.7±4.3	Y

Table IV: Mean absolute error for variance query.

	Ideal Local DP		FxP HW Baseline		Resampling		Thresholding	
Dataset	MAE	LDP?	MAE	LDP?	MAE	LDP?	MAE	LDP?
Auto-MPG	39±2.6 (9.9%)	Y	40±2.5	N	29±2.7	Y	40±2.5	Y
Robot Sensors	867±15 (14%)	Y	868±15	N	867±15	Y	868±15	Y
Statlog (Heart)	25±2.0 (9.2%)	Y	25±2.0	N	25±2.0	Y	25±2.0	Y
Human Activity	2370±47 (6.8%)	Y	2377±47	N	2389±47	Y	2377±47	Y
Localization for Person	1245±26 (4.3%)	Y	1244±26	N	1927±0.0	Y	1244±26	Y
UJIIndoorLoc	744±24 (3.7%)	Y	744±24	N	747±23	Y	744±24	Y
Postural Transitions	421±25 (1.8%)	Y	421±25	N	429±25	Y	421±25	Y

Table V: Mean absolute error for counting query.

fourth settings, resampling and thresholding were added on the baseline to guarantee differential privacy. The MAE values shown in the tables are MAE $\pm$ standard deviation of MAE. Basically the values in the tables show how much query outputs calculated from DP outputs are off from the exact query output obtained from raw data. We also showed the relative error normalized to the full data range.

Although, as shown in the tables, the proposed DP-box with resampling and thresholding shows similar utility result for all the queries and datasets as that of ideal local DP case, there are several things to note in the result. First, FxP hardware baseline always shows almost identical utility results with ideal distribution for all the queries and for all the datasets, which implies that low-power FxP hardware can perform as well as ideal case. However, all the baselines suffer from infinite privacy loss as described in the previous section, and they cannot guarantee DP. On the other hand, the proposed DP-box uses resampling or thresholding to guarantee DP at all times, but each changes the added noise distribution differently as shown in Fig. 6 and Fig. 7. Moreover, the noise distribution becomes different based on the original sensor data while ideal case and baseline add the same noise distribution ($\mathrm{Lap}(d/\varepsilon)$) regardless of sensor data. Thus, the utility of resampling and thresholding depends on the distribution of a given dataset. For example, for a dataset that has many elements near the boundary (minimum or maximum of the sensor data), e.g. the dataset of Localization for Person, the utility with resampling becomes much worse than ideal case, while

that with thresholding performs similar to the ideal case. On the other hand, for many datasets having Gaussian-like distribution (many elements located around the mean), resampling shows better utility. In summary, generally resampling and thresholding show similar utility to ideal distribution, but their utility depends highly on the data distribution, so it is required to consider the underlying data distribution and the application carefully before applying local DP to the sensor data.

Figure 11 shows the corresponding latency values (in cycles). Recall that DP-Box takes 2 cycles when thresholding is used. Every re-sample adds one cycle. Results show that resampling never adds more than a cycle, on average (often much lower). This demonstrates that DP-Box can provide privacy guarantees at low latency overhead.

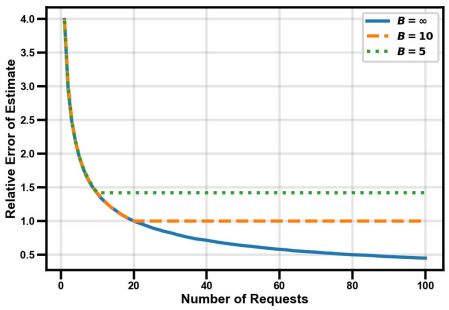

Figure 13: Comparison of the estimate accuracy with three different values for privacy budget.

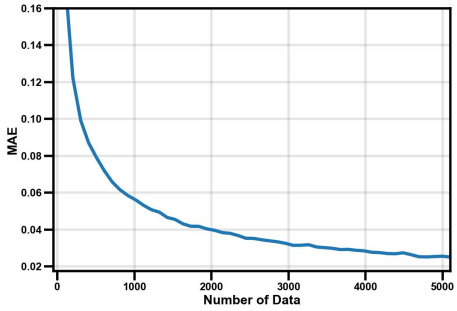

Figure 14: Simulated mean absolute error of randomized response implemented in the proposed DP-box.

C. Utility Sensitivity to Dataset Sizes and RNG Resolution

For the queries having global sensitivity scaled with the number of data such as mean, variance, or standard deviation, the accuracy of the query on noised data becomes better as the number of data entries increases. Figure 15 shows the simulated MAE for the mean query as a function of the number of data entries. As shown in Fig. 15(a), if the number of RNG output bits are sufficiently large, then MAE of all four settings show similar utilities and error approaches zero as the number of data increases, thus enabling more accurate aggregate statistics collection while preserving each data privacy. However, if the number of RNG output bits is small,

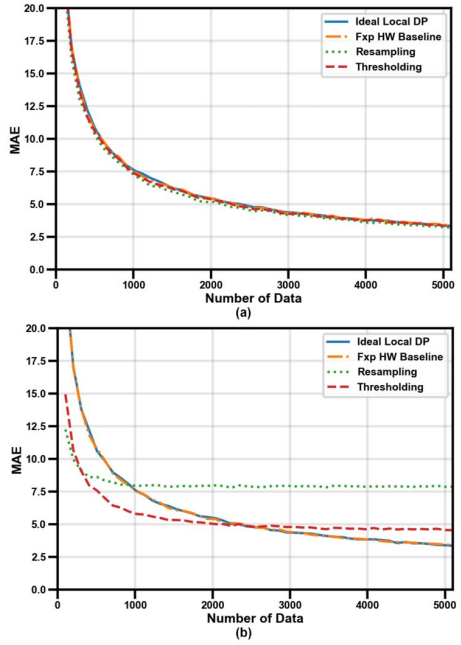

Figure 15: Simulated mean absolute error as a function of number of data: (a) $\varepsilon = 0.5$ and 24 bits of URNG output and (b) $\varepsilon = 0.5$ and 16 bits of URNG output.

then resampling and thresholding set the threshold small and the resulting noise distribution becomes much different from the ideal Laplace distribution. Due to this, even if the number of data increases, MAE does not approach zero and there exists a lower limit as shown in Fig. 15(b).

D. Effectiveness of Privacy Budget Control

Finally, in order to demonstrate that the proposed privacy budget control algorithm allows DP-box to guarantee local privacy even in presence of an adversary, we performed the following experiment. When an adversary wants to reveal the original sensor data, the adversary will request sensor data multiple times to the DP-box until the budget is consumed up and then take the average of the noised output samples, which is the maximum likelihood estimate of the original sensor data. In this setting, we compare three cases (Fig. 13). In one case, there is no privacy budget so there is no limit on the number of requests that the adversary can send. In the other two cases, we set the privacy budget to two different values for comparison. The DP-box provides the noised outputs with $\varepsilon = 0.5$ in all three cases and, with finite privacy budget, when all the privacy budget is consumed, the last cached output is provided as DP output. Figure 13 shows the relationship between the relative error of the sensor data estimate and the number of data requests. As can be seen, without privacy budget control logic in the DP-box, the accuracy of the estimate keeps improving as the number of requests increases, and the error will eventually approach zero. On the other hand, with finite privacy budget, it is possible to limit the accuracy

of the adversary's estimate, which shows the effectiveness of the privacy budget control logic in the DP-box.

E. Using DP-Box to Support Randomized Response

The Laplace mechanism is not the only technique that enables collecting user data while preserving their privacy. In statistics, randomized response techniques [19] have been extensively investigated for collecting categorical data. For example, Google introduced a new mechanism called RAPPOR [40] based on the randomized response technique that is performed locally on the client side and that does not require a trusted third party. The proposed DP-box can be reconfigured to support the randomized response mechanism by setting the threshold zero. Thus, it can be used to handle not only numeric data from sensors but also categorical data. For example, if we want to keep the gender information (male or female) of the users in Statlog heart-rate dataset private, the proposed DP-box with thresholding while setting the threshold zero implements the randomized response technique, i.e. the data and the noised output are both binary. In order to measure the utility, the mean absolute error of the population of male in the dataset was simulated when the randomized response technique was applied. As shown in Fig. 14, the query accuracy becomes more accurate as the number of data increases while preserving the privacy of individual data.

F. Application to Privacy-Preserving Learning Tasks

In the previous section, it was shown that aggregate data statistics can be obtained while preserving individual data privacy. In addition to data statistics, local differential privacy can also be potentially used for many machine learning applications. For example, classification models such as logistic regression or support vector machine (SVM) can be trained while preserving data privacy using DP [41].

Data Size	1000	2000	3000	4000	5000
$\varepsilon = 0.5$	69%	72%	76%	77%	82%
$\varepsilon = 1$	79%	82%	85%	87%	90%
$\varepsilon = 2$	87%	90%	91%	93%	94%
No DP	96%	98%	98%	99%	99%

Table VI: Classification accuracy with different training set size and privacy parameter.

In order to demonstrate this, we generated a synthetic dataset for binary classification, which is separable by a halfspace. Since the data is separable, when we train an SVM using a training set, the classification accuracy approaches 100% as the training data size becomes larger. Performance of the SVM trained using true data was compared with those trained using noised data with different privacy parameters in Table VI. The final classification accuracy was tested with identical test data set without noise in all cases. As shown in Table VI, the SVM can be trained even with noised data, but the number of data needed to achieve the same performance becomes larger as the data becomes more private (smaller ε), which is the cost we pay for preserving privacy.

VII. Conclusion

One fundamental challenge for future sensor and IoT systems is how to draw meaningful information from sensor data while maintaining the privacy of the data and individuals. In this work, we explored the feasibility of providing local differential privacy on ULP systems that power many sensor and IoT applications. We showed that low resolution and fixed point nature of ULP implementations prevent privacy guarantees from being provided due to low quality noising. We presented techniques, resampling and thresholding, to overcome this limitation. The techniques, along with a privacy budget control algorithm, were implemented in hardware to provide privacy guarantees with high integrity. We showed that hardware implementation, DP-Box, has low overhead and provides high utility, while guaranteeing local differential privacy, for a range of sensor/IoT benchmarks.

Acknowledgment

This work was supported in part by Analog Devices and Silicon Labs.

References

[1] M. Magno, L. Benini, C. Spagnol, and E. Popovici, "Wearable low power dry surface wireless sensor node for healthcare monitoring application," in *IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 189–195, 2013.

[2] C. Hierold, B. Clasbrumme, D. Behrend, T. Scheiter, M. Steger, K. Oppermann, H. Kapels, E. Landgraf, D. Wenzel, and D. Etuodt, "Implantable low power integrated pressure sensor system for minimal invasive telemetric patient monitoring," in *Annual International Workshop on Micro Electro Mechanical Systems*, pp. 568–573, 1998.

[3] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *International Conference on Embedded Networked Sensor Systems*, pp. 13–24, 2004.

[4] H. W. Gellersen, A. Schmidt, and M. Beigl, "Multi-sensor context-awareness in mobile devices and smart artifacts," *Mobile Networks and Applications*, vol. 7, no. 5, pp. 341–351, 2002.

[5] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications magazine*, vol. 48, no. 9, 2010.

[6] V. Pejovic and M. Musolesi, "Anticipatory mobile computing: A survey of the state of the art and research challenges," *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, p. 47, 2015.

[7] K. J. Cios and G. W. Moore, "Uniqueness of medical data mining," *Artificial intelligence in medicine*, vol. 26, no. 1-2, pp. 1–24, 2002.

[8] R. Dewri, "Local differential perturbations: Location privacy under approximate knowledge attackers," *IEEE Transactions on Mobile Computing*, vol. 12, no. 12, pp. 2360–2372, 2013.

[9] G. Danezis, M. Kohlweiss, and A. Rial, "Differentially private billing with rebates," in *International Workshop on Information Hiding*, pp. 148–162, Springer, 2011.

[10] B. Mobasher, R. Cooley, and J. Srivastava, "Web usage mining can help improve the scalability, accuracy, and flexibility of recommender systems," *Communications of the ACM*, vol. 43, no. 8, pp. 142–151, 2000.

[11] P. K. Chan, W. Fan, A. L. Prodromidis, and S. J. Stolfo, "Distributed data mining in credit card fraud detection," *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 6, pp. 67–74, 1999.

[12] J. C. Herrera, D. B. Work, R. Herring, X. J. Ban, Q. Jacobson, and A. M. Bayen, "Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 4, pp. 568–583, 2010.

[13] S. Pearson, "Taking account of privacy when designing cloud computing services," in *Software Engineering Challenges of Cloud Computing, 2009. CLOUD'09. ICSE Workshop on*, pp. 44–52, IEEE, 2009.

[14] C. Dwork and A. Smith, "Differential privacy for statistics: What we know and what we want to learn," *Journal of Privacy and Confidentiality*, vol. 1, no. 2, p. 2, 2010.

[15] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *Annual International Cryptology Conference*, pp. 528–544, Springer, 2004.

[16] B. R. Kandukuri, V. R. Paturi, and A. Rakshit, "Cloud security issues," in *Proceedings of the 2009 IEEE International Conference on Services Computing*, pp. 517–520, IEEE Computer Society, 2009.

[17] F. X. Lin, Z. Wang, R. LiKamWa, and L. Zhong, "Reflex: using low-power processors in smartphones without knowing them," *ACM SIGARCH Computer Architecture News*, vol. 40, no. 1, pp. 13–24, 2012.

[18] "International technology roadmap for semiconductors 2.0." http://www.itrs2.net, 2015. [Online].

[19] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[20] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*, pp. 265–284, Springer, 2006.

[21] Q. Geng and P. Viswanath, "The optimal noise-adding mechanism in differential privacy," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 925–951, 2016.

[22] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[23] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *Advances in neural information processing systems*, pp. 2879–2887, 2014.

[24] P. Hellekalek, "Good random number generators are (not so) easy to find," *Mathematics and Computers in Simulation*, vol. 46, no. 5-6, pp. 485–505, 1998.

[25] R. C. Cheung, D.-U. Lee, W. Luk, and J. D. Villasenor, "Hardware generation of arbitrary random number distributions from uniform distributions via the inversion method," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 8, pp. 952–962, 2007.

[26] C. De Schryver, D. Schmidt, N. Wehn, E. Korn, H. Marxen, A. Kostiuk, and R. Korn, "A hardware efficient random number generator for nonuniform distributions with arbitrary precision," *International Journal of Reconfigurable Computing*, vol. 2012, p. 12, 2012.

[27] I. Mironov, "On significance of the least significant bits for differential privacy," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 650–661, ACM, 2012.

[28] I. Gazeau, D. Miller, and C. Palamidessi, "Preserving differential privacy under finite-precision semantics," *Theoretical Computer Science*, vol. 655, pp. 92–108, 2016.

[29] S. Chakraborty, K. R. Raghavan, M. P. Johnson, and M. B. Srivastava, "A framework for context-aware privacy of sensor data on mobile systems," in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, p. 11, ACM, 2013.

[30] M. Maycock and S. Sethumadhavan, "Hardware enforced statistical privacy," *IEEE Computer Architecture Letters*, vol. 15, no. 1, pp. 21–24, 2016.

[31] A. U. Schmidt, N. Kuntze, and M. Kasper, "On the deployment of mobile trusted modules," in *Wireless Communications and Networking Conference*, pp. 3169–3174, IEEE, 2008.

[32] U. NXP, "I2C-bus specification and user manual," 2007.

[33] M. Lichman, "UCI Machine Learning Repository," 2013.

[34] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, 2016.

[35] T. Oates, M. Schmill, and P. R. Cohen, "Identifying qualitatively different experiences: Experiments with a mobile robot," 2000.

[36] J. R. Quinlan, "Combining instance-based and model-based learning," in *Proceedings of the tenth international conference on machine learning*, pp. 236–243, 1993.

[37] B. Kaluža, V. Mirchevska, E. Dovgan, M. Luštrek, and M. Gams, "An agent-based approach to care in independent living," in *International joint conference on ambient intelligence*, pp. 177–186, Springer, 2010.

[38] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta, "UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems," in *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*, pp. 261–270, IEEE, 2014.

[39] W. Ugulino, D. Cardador, K. Vega, E. Velloso, R. Milidiú, and H. Fuks, "Wearable computing: Accelerometers data classification of body postures and movements," in *Advances in Artificial Intelligence-SBIA 2012*, pp. 52–61, Springer, 2012.

[40] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 1054–1067, ACM, 2014.

[41] K. Chaudhuri and D. Hsu, "Sample complexity bounds for differentially private learning," in *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 155–186, 2011.