This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2015.2403359, IEEE Computer Architecture Letters

Hardware Enforced Statistical Privacy

Matthew Maycock Simha Sethumadhavan
Computer Architecture and Security Technology Lab (CASTL)
Department of Computer Science
Columbia University, NY 10027
{mhm2159, simha}@columbia.edu

Abstract—The Internet of Things will result in users generating vast quantities of data, some of it sensitive. Results from the statistical analysis of sensitive data across wide ranges of demographics will become ever more useful to data analysts and their clients. The competing needs of the two groups—data generators with their desire for privacy and analysts with their desire for inferred statistics—will be met through the use of statistical privacy techniques. The question, then, is how can we ensure that the statistical methods are applied in a trustable manner? In this paper we discuss some of the complications and consequences of ensuring both trust and privacy through the immutability of hardware, providing a desiderata for a hardware privacy platform.

Index Terms—Internet of Things, Privacy, Hardware Support, Privacy Protection Unit

1 Introduction

Today many ordinary, day-to-day devices have embedded radios and computational capabilities that can either be remotely controlled or operate autonomously. These devices collect, process, store, and respond to the data collected from the world around them. This general trend towards enabling connectivity and autonomy of simple devices is often referred to as the Internet of Things (IoT) in popular literature. With IoT, an individual's own devices interact not only with each other, but also with third parties: device manufacturers, emergency medical services, and mobile phone service providers, among others. In these systems it is vital to balance users' privacy needs with business requirements. In this paper we discuss if and how computer architects can help with addressing privacy concerns in this interconnected (IoT) world.

The third parties in IoT may be trusted to different degrees. Ideally we would like fully trusted third parties to get access to the original data, provide degraded data to partially trusted third parties, and completely deny data access to untrustworthy parties. The mechanism for the first and the last use cases is straightforward. Any mechanism for providing degraded data to a partially trusted third party must guarantee that the original data cannot be easily recovered from the degraded data. This guarantee requires careful mathematical reasoning and is the subject of many studies in the field of statistical privacy. One particular theory proposed in 2006, ϵ-differential privacy, offers a rigorous mathematical foundation and comes with a simply stated privacy guarantee [1]. The idea is to add noise in such a way that the (statistical) usefulness of results is preserved while the ability to infer the value, or even presence or absence, of an individual record is deterred.

What is the benefit of hardware support for ϵ-differential privacy? The cost of applying differential privacy is not significant enough to warrant hardware support: the computation amounts to a few memory accesses and arithmetic operations to add noise at the end of a computation. Hardware immutability can, however, provide more substantial privacy assurances to developers and end users than software mechanisms can. Like Trusted Platform Modules for security, we envision a hardware Privacy Protection Unit (PPU) that mediates access to sensitive sensor data and provides ISA abstractions to enable easy development of privacy-aware systems. Providing hardware based privacy mitigates any danger resulting from coverage holes, which may arise due to the complexity of software (interactions). Further, when combined with hardware security mechanisms such as isolation and attestation, our PPU abstractions can provide proofs of privacy that cannot be achieved easily with purely software implementations. As a first step towards privacy-aware systems with hardware-rooted privacy, in this paper we describe the PPU architecture, discuss desiderata for a hardware-rooted privacy policy enforcement engine (§3.2.1), and discuss new ISA primitives (§3.2.2, 3.2.3) that provide assurance that privacy has been correctly applied in hardware, leaving quantitative evaluation to future work.

The rest of the paper is organized as follows: In Section 2 we provide background on statistical privacy. Section 3 discusses privacy facilitating features architects can provide. We conclude in Section 4.

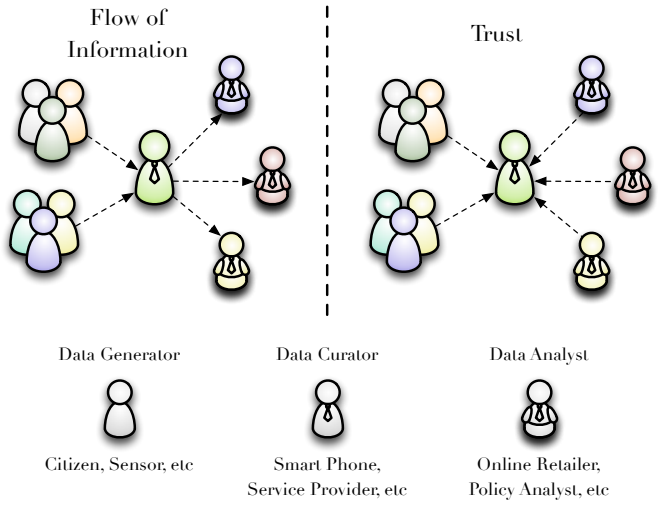

Fig. 1. Trust & Information Relationships

2 Background

Figure 1 shows an abstract view of trust relationships in an IoT ecosystem. IoT actors can be abstracted into three entities: data generators—e.g. a heart rate sensor; data curators—mobile phone vendors, third party entities, etc.; and one or more data analysts with differing goals and authorizations to the sensitive data. Statistical privacy measures are applied by the data curator to prevent (some) data users from reconstructing

. This work is supported through grant FA8750-10-2-0253 and the Alfred P. Sloan Foundation. Opinions, findings, conclusions and recommendations expressed in this material are those of the authors and may not reflect the views of the funding entities.

. Manuscript submitted: 09-Dec-2014. Manuscript accepted: 11-Jan-2015. Final manuscript received: 16-Jan-2015

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2015.2403359, IEEE Computer Architecture Letters

the original data from the generator. The trust relationships and aims of the different actors are best explained with an example.

Consider the case of a mobile phone service provider that, through sensors on both phones and other devices (e.g. bluetooth fitness tracker), has historical location and user habit information for its customers. Such information can reveal detailed personal information: which shops users visit (and when) or even what activities usually precede shop visits (e.g. smoothies after exercise). This information, combined with demographic information available to the service provider, can be used to build very accurate shopping and lifestyle profiles.

This information has many uses. For example, it could be used by an online retailer to get very useful information on where locals are looking for goods. If such a retailer desired to provide same day delivery, such information could be used to reduce delivery times, increasing the retailer's competitive edge against brick and mortar stores. The emergence of IoT promises very fine-grained, precise information that will help retailers optimize these tasks even further.

The above scenario involves subtle relationships that need to balance trust and utility: if the mobile service provider gives away information in a manner that betrays the public trust, they may lose many customers (becoming less useful to analysts, losing revenue streams on two fronts); if the provider obfuscates the information too greatly the retailer will lose interest. The provider needs to account for both the public's (customers') privacy needs and the analyst's (retailer's) statistical needs. In the rest of the section we will describe statistical techniques for achieving these goals.

2.1 Differential Privacy

Let f be some computation on datasets, e.g. f returns mean salary. The goal of differential privacy is to prevent analysts from inferring the values, or even presence, of individual records from the values $f(D_1)$ and $f(D_2)$ where D_1 and D_2 are neighboring datasets. Two datasets are neighbors if they differ by the absence/presence of a single record.

In her seminal work [1], Dwork defines a noised (i.e. randomized) mechanism $\mathcal{K}_f$—parameterized on the computation f—as being ϵ-differentially private if the addition or omission of a single record is unlikely to change the result significantly.[1]

Dwork shows that adding double symmetric noise (i.e. Laplacian) to a calculation yields an ϵ-differentially private mechanism. The noise is centered with location zero and has scale based on a privacy parameter and the computation. [2]

We note that privacy measures such as these induce a trade-off between integrity and confidentiality. Normally confidentiality is binary property—either an entity can read the data or it cannot. The privacy derived from noising sensitive information turns this binary property into a continuous property: adversarial access to information increases as privacy decreases. For example, location data in the earlier mobile phone example can be seen as confidential information; however as we lower privacy the accuracy of locating the user increases and thus becomes 'less' confidential.

1. $\Pr[K_f(D_1) \in S] \leq \exp(\epsilon) \times \Pr[K_f(D_2) \in S]$ where S is any given subset of the range of K_f and D_1, D_2 are neighboring datasets.

2. The scale is $\Delta f/\epsilon$. ϵ is the privacy parameter: smaller values yield wider noise and thus greater privacy. Δf is the maximum change of f among neighboring data sets, e.g $\Delta f = 1$ if f merely counts records. Larger Δf yields narrower noise and thus less privacy.

TABLE 1
Common Identity & Attribute Protection Schemes

Privacy Method	Summarization
k-anonymity	Each generalized quasi-identifier's equivalence class has at least k elements. **Offers**: Identity protection.
l-diversity	Each generalized quasi-identifier's equivalence class has l "well-represented" values for each sensitive attribute. **Offers**: Identity protection and *some* attribute protection that is still statistically vulnerable.
t-closeness	Each generalized quasi-identifier's equivalence class's sensitive attribute values are distributed similarly to the overall table. **Offers**: Identity protection along with reasonable attribute protection.

2.2 Other Forms of Privacy

Other forms of statistical privacy exist. We discuss three related methods below (summarized in Table 1). First we require a definition: a *quasi-identifier* is a set of attributes that collectively identify an individual. Examples include the combination of age, sex, and postal code, which when cross-referenced against voter registration can identify an individual [6].

k-anonymity [7] requires that a published table, cleared of explicit identifiers (names, social security numbers, etc), generalize quasi-identifiers so that quasi-identifier equivalence classes have at least k entries. The method offers identity level privacy: we do not know which record corresponds to a specific individual. The scheme, however, does not offer attribute level privacy: if we know which equivalence class contains individual X and every member of that class has the same disease, then we know that X has the given disease.

To combat the above limitation of k-anonymity, and others based on background information, Machanavajjhala, et al. [3] instead propose the concept of l-diversity. l-diversity requires that each class of individuals have a diverse set of values for each sensitive attribute. For example, not only there should be no class where every member has cancer but there should not be classes where every member has either cancer or heart disease—as then knowing that member X is from a demographic with a low incidence of heart disease allows us to infer that X likely has cancer. There are many different ways to measure and thus enforce l-diversity, such as requiring each class to have at least at least l distinct values or the total entropy of values in a class be at least $\log l$.

l-diversity, while solving issues with k-anonymity, comes with its own issues. As described by Li and Li [2], if a disease only affects a small percent of the population but an equivalence class highly represents those with the disease, then a great deal of information can be gained. What t-closeness proposes is limiting the distance (of distributions) between the overall distribution of sensitive values in the overall table and the distribution of sensitive values within equivalence classes.

3 Privacy & Computer Architecture

As computer architects, we examine how hardware support can assist with statistical privacy. We consider the low-level but general case that the data generators are sensors—an ambient light sensor, a gyroscope, a compass, a GPS unit, even a camera—and that the analysts are processes requesting access

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2015.2403359, IEEE Computer Architecture Letters

to the sensitive data available from the sensors. We present a platform that acts as what we have called a data curator, and examine the following aspects related to statistical privacy: (1) What are the benefits of hardware privacy support? (2) What privacy services can be offered by hardware?

3.1 Benefits of Privacy Support

Common benefits for supporting a computation with hardware are lower latency and energy. Another common benefit is trust due to the immutability of hardware: software can easily be corrupted and subverted, but validated hardware must be *physically* corrupted in order to be subverted.

What are the benefits of hardware support for privacy? Noise generation is computationally inexpensive—both in terms of latency and energy efficiency since it is a single calculated value added to a signal or computational result. We can, however, use hardware to provide a robust, difficult to subvert *privacy platform* with very little software support. Such a platform could include features facilitating applications' use of sensitive data while assuring users that their data is not being misused.

3.2 Privacy Protection Platform

The privacy methods we have examined fall into two camps: noising computational results (ϵ-differential privacy, etc) and partitioning records according to a rule while generalizing quasi-identifiers (k-anonymity, l-diversity, t-closeness, etc). The latter methods are highly dependent on the data attributes and general-purpose ISA support is likely to be infeasible.

Regarding noising, there are two possible methods: noising at the source (a sensor), or at the sink (a computational use within a program). Though these two options are related—in fact, mathematically speaking, noising inputs is equivalent to post-processing the noised output of the identity function—they lead to very different system designs. Ny and Pappas [5] discuss the trade-offs of applying ϵ-differential privacy in these two ways in the context of application-specific dynamic systems taking Kalman filters as an example. We explore these options in a general-purpose SoC setting with emphasis on programmability, forensics, and robustness.

Noising the data at the sinks requires information flow tracking: sensitive data should be tainted and all operations on tainted data should either be restricted (i.e. exfiltrating tainted data should be disallowed), yield a tainted result, or be a special detainting operation which should trigger policy enforcement. Such a system would need to track all of the different sources of tainted data requiring even a single byte to carry around a large amount of metadata. The policy and mechanism design is genuinely complicated in this case while both the energy and performance overheads can be substantial.

Noising at the source, however, requires minimal changes to the hardware and software. An off-chip unit that sits between the sensors and the processor can mediate access to sensitive sensor data and control how noised data is supplied to the processor. The off-chip unit, which we call the privacy protection unit (PPU), can provide data at different privacy levels to processes based on a policy set by a trusted executive. The trusted executive can be much simpler to implement than a sink enforcement policy due to centralized policy enforcement.

Figure 2 gives an overview of our privacy additions. In the figure, the PPU accepts two different types of requests: policy management requests from the trusted executive and sensor read requests from processes. Sensor read requests are used to access sensitive data in a controlled manner. Requests first arrive at an access mediation unit which acts as a gate-keeper, ensuring that only allowed requests are processed. Next the PPU reads from the appropriate sensor, passing the value on to the noise addition and budget control unit, which validates that the request is within budget and appropriately noises the result. Finally, the access and noise logging unit logs both the noise and the access, providing a record of privacy. All of these steps are configured through policy management requests sent by the trusted executive.

Each component is capable of raising an exception; additionally, policy management requests can raise exceptions. The access mediation unit raises an exception when the request is disallowed by the PPU policy. The noise addition and budgeting control unit can raise an exception when the requested level of privacy is either invalid (≤ 0) or greater than the remaining budget. The access and noise logging unit raises its exception when one of the logs fills up. Finally, policy management requests can cause exceptions if the request contains illegal policy programming or is malformed in some fashion.

We now describe the primary components of the PPU.

3.2.1 Policy Enforcement

The policy enforcement engine can be a fixed function unit supporting a single policy (for embedded IoT devices) or a programmable/configurable unit to allow a range of policies. Irrespective of these choices the policy enforcement engine has to be resilient to intentional or unintentional privacy attacks. In this section we describe one attack and derive the desiderata for the policy enforcement engine.

A collusion attack. A system with repeated querying must consider the problem of collusion among entities spread among different user accounts, different applications across the same or several user accounts, or even the same application running as several different processes. For example, assume that location is meant to be kept private and, within a relatively short period of time, n applications—possibly on the same user account, possibly not—query the GPS unit with a modest amount of privacy. Then, these n applications can average their results to get a result whose privacy is less than modest. This breach results from the fact that the noise can be averaged out.

Budgeting solution. Limiting users, groups, applications, or even the entire system—or non-colluding categories thereof—to privacy budgets can alleviate the above privacy breach. In the parlance of ϵ-differential privacy, the budget can be the total sum of ϵ values of executed queries on related sources. As explained by Dwork [1] the privacy expenditure of multiple ϵ-differentially private queries is at worst additive in the privacy parameter, making this an acceptable solution. Such budgeting solutions have been used in privacy-aware querying systems at the software level [4].

The privacy budget can be system wide, source specific, or anything in between and allocated for a programmable duration of time. However, holistic system-wide notions of privacy are desirable since if two sensors yield related information, then queries to one must affect the privacy budget of the other. For instance, location information can be obtained via GPS, wi-fi strength, or cell tower information. An additional requirement is that the policy enforcement engine should support replenishment, i.e. how, over time, is the privacy budget restored, if at all? Several policies are possible: the replenishment may be based on elapsed time, sensor information such as location, or some other arbitrary policy from the executive. A full system

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2015.2403359, IEEE Computer Architecture Letters

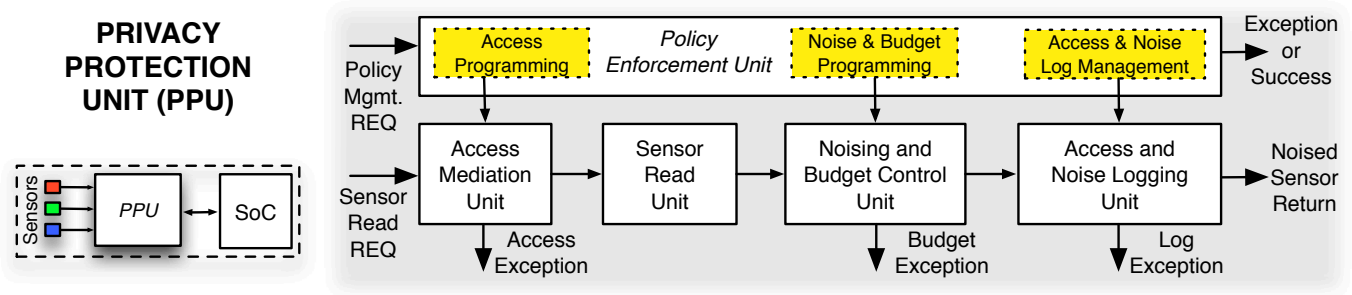

Fig. 2. Privacy Platform Overview

design along with an implementation is essential to understand the relations between the sensor uses, budgets, and utility.

Importantly, we note that these are *privacy* policy decisions, not *security* policy decisions—if a trusted process (i.e. one with a significantly sized budget) publishes results or is poorly programmed and leaks information, then these are important but separate issues from what we address here.

3.2.2 Assurance of Noise

It is desirable for users to be assured that their privacy needs are being met. Towards this goal we propose a hardware structure to provide assurance of noise. To do so, the PPU maintains a hardware unit with a buffer of recently calculated noise values. If, given the observed distribution, the likelihood of the desired distribution was sufficiently small, the access and noise logging unit could raise a log exception.

If queries only add noise from the same class of distribution, either with the same or different parameterizations, then the system can translate noise values to their canonically parameterized distribution values—i.e. the normal distribution with $\mu = 0, \sigma = 1$ in the case of Gaussian noise. Comparing these observed values against the desired distribution allows the system to provide assurance of noise across different queries, requiring only one buffer for each class of noise distribution.

3.2.3 Assurance of Access

In addition to the above, the platform could provide assurance that a user or application accessed certain data with a given privacy level. Just as modern operating systems come with "task managers" and "activity monitors," an assurance of access unit could provide data for a user friendly privacy manager. Taken together with an assurance of noise unit, this could be a very powerful tool: if privacy breaks down for some reason—i.e. noise is not properly distributed—then users will be able to inspect whether certain apps took advantage of the situation and decide if such apps constitute malware.

Assurance of access inherently requires logging and is thus vulnerable to denial of service attacks. Imagine that a user accesses a piece of data with relatively low privacy (i.e. large ϵ), but then other users—or even the same user—access that data and other data with a great deal of privacy (i.e. small ϵ). If each access is logged as an individual event and there is only room for a fixed number of events, then the original access may be overwritten. Virtual memory like solutions or clever logging techniques would be necessary to avoid such pitfalls in the implementation of an assurance of access mechanism.

The above two assurance mechanisms, taken together with security techniques such as trusted platform modules, can form a proof of privacy. That is to say, the assurances, accompanied with proof of their authenticity, themselves become proofs in the cryptographic sense.

4 Conclusions

The emergence of IoT brings about important privacy questions. Computer architects typically optimize for the common case and the demand to balance privacy with business needs in the IoT is likely to be a fairly common use case. Thus, in this paper we examined the feasibility and benefits of hardware support for privacy in IoT devices. Our study has revealed new ISA primitives for privacy, viz., instructions that not only provide privacy aware access to sensors but also verify privacy ands its use within the system. We described an implementation in which the privacy control is performed as soon as sensitive data is acquired and before any further processing of the data. In addition to reducing the risk of sensitive data exposure, such systems are also much easier to implement than systems that enforce privacy upon exfiltration to non-privacy aware (hardware or software) components. We discussed the nuances of policy enforcement for our privacy protection unit with one specific policy example. Based on this initial study we conclude that much interesting work remains to be done in terms of hardware support for privacy. Specifically full system prototypes are necessary to understand the utility of hardware supported differential privacy in real-world use cases.

References

[1] C. Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12, Venice, Italy, July 2006. Springer Verlag.

[2] N. Li and T. Li. t-closeness: Privacy beyond k-anonymity and l-diversity. In *In Proc. of IEEE 23rd Int'l Conf. on Data Engineering (ICDE '07)*, 2007.

[3] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.

[4] 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. For more information, visit the project page: http://research.microsoft.com/PINQ.

[5] J. L. Ny and G. J. Pappas. Differentially private filtering. In *CDC*, pages 3398–3403. IEEE, 2012.

[6] L. Sweeney. *Computational Disclosure Control: A Primer on Data Privacy Protection*. PhD thesis, Massachusetts Institute of Technology, 2001. AAI0803469.

[7] L. Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, October 2002.