A Cloud-Based Consumer-Centric Architecture for Energy Data Analytics

Rayman Preet Singh, S. Keshav, and Tim Brecht
{rmmathar, keshav, brecht}@uwaterloo.ca
School of Computer Science, University of Waterloo,
Waterloo, Ontario, Canada

ABSTRACT

With the advent of utility-owned smart meters and smart appliances, the amount of data generated and collected about consumer energy consumption has rapidly increased. Energy usage data is of immense practical use for consumers for audits, analytics, and automation. Currently, utility companies collect, use, share, and discard usage data at their discretion, with no input from consumers. In many cases, consumers do not even have access to their own data. Moreover, consumers do not have the ability to extract actionable intelligence from their usage data using analytic algorithms of their own choosing: at best they are limited to the analysis chosen for them by their utility. We address these issues by designing and implementing a cloud-based architecture that provides consumers with fast access and fine-grained control over their usage data, as well as the ability to analyse this data with algorithms of their choosing, including third party applications that analyse that data in a privacy preserving fashion. We explain why a cloud-based solution is required, describe our prototype implementation, and report on some example applications we have implemented that demonstrate personal data ownership, control, and analytics.

Categories and Subject Descriptors

C.0 [**Computer Systems Organization**]: General—*System architectures*; D.2.11 [**Software**]: Software Architectures

Keywords

Home energy, data privacy, data analytics, third party applications, system architecture

1. INTRODUCTION

Utilities around the world are deploying "smart meters" to record and report energy consumption readings to utility central offices. This enables different prices to be charged for electricity based on the time of day and eliminates the cost of a monthly visit by a meter reader. The time series of meter readings, originally meant only for customer billing, has unanticipated uses. On the one hand, customers who have access to their usage data can get a real-time, fine-grained view into their electricity consumption patterns. When suitably analysed, this can reveal potential cost savings and customized guidance on the benefits from energy conservation measures, such as installing insulation, solar panels, or purchasing energy-efficient products. On the other hand, this same data stream can reveal private information about the customer, for example, when they are home and when they are not, the appliances they own, and even, in some cases, which TV channel or movie they are watching [30, 39]!

Unlike traditional utility-centric approaches to data management in the smart grid, we instead take a consumer-centric approach [37]. We believe that consumers would like to:

- have control over their own data while outsourcing data storage and persistence to an infrastructure provider
- get a single view into data collected from multiple sources
- give access to their data to analytic algorithms of their choice, but without giving up data privacy

These goals are not achieved by any existing solution. Today, many utilities do not even provide consumers with access to their own usage data. Even the utilities that give consumers access to data, such as those participating in the Green Button initiative [6], or those that provide rudimentary analytics, do not allow consumers to use analytic algorithms of their own choosing. Finally, no current system gives consumers fine-grained control over who can access the data, and the granularity and period of time at which it can be accessed.

Building on the rich infrastructure of modern clouds, we have designed and implemented cloud-based personal data and execution containers that persistently store data and offer an environment for the execution of arbitrary analytic algorithms. Consumers can use these containers to grant fine-grained access to their data to third parties. These containers also allow secure and private control of home appliances from any Internet-enabled device.

The key contributions of our work are:

- The design of a system that allows consumers to own and control access to their energy usage data and have it analysed using algorithms of their choice

e-Energy'13, May 21–24, 2013, Berkeley, California, USA.
Copyright 2013 ACM 978-1-4503-2052-8/13/05 ...$15.00.

- A proof-of-concept implementation of our architecture on modern cloud computing platforms
- An evaluation of the system architecture with respect to data access and control

The remainder of the paper is organized as follows. Section 2 describes related research. Section 3 outlines the goals and requirements of our system and Section 4 explains the rationale for our architecture. Section 5 presents a detailed description of the architecture followed by a description of our prototype implementation in Section 6. We evaluate our architecture in Section 7, discuss practical implications in Section 8 and limitations in Section 9. Our conclusions are presented in Section 10.

2. RELATED WORK

We group related work into the following categories: Personal Data, Energy Data, Energy Data Privacy and Systems Architectures.

Personal Data: Researchers have proposed ecosystems built around an individual's data, such as health records, smart meter data, data concerning banking, taxation and shopping [41]. McAuley et al. [38] introduce the concept of *dataware* that defines the processes of obtaining, accessing and using an individual's data. Haddadi et al. [31] report that the ethical and legal consequences of gathering individuals' data are not yet fully determined, but it is understood that the individual co-owns any data concerning them. We focus on applying the concept of *privacy-preserving dataware* to an individual's energy data and investigate the goals, architecture, and mechanisms needed to implement such a system.

Energy Data: Currently, various utility and software companies provide consumers with access to their energy data through web portals. Examples include initiatives like Green Button [6], analytics providers like Opower [10], utility companies such as Waterloo North Hydro [14], San Diego Gas and Electric [13], and software projects like Google Powermeter [5] and Microsoft Hohm [8] (both now defunct). While these portals allow residential and commercial consumers to download data about their energy consumption (or *energy data*), the consumer is responsible for its long term storage and use. In many cases, the data is only available for a limited time (e.g., three months [14]) and hence such portals do not provide consumers with a durable storage solution. Secondly, the data analytics available to consumers is at the utilities' discretion. As a result they are deprived of potentially better analytics through third-party applications. We focus on building a platform to circumvent these problems.

Energy Data Privacy: Analysing smart meter data in a privacy-preserving fashion has been the focus of much research. Most work has focused on applying obfuscation, aggregation and homomorphic encryption to energy data [18,29,35,36,47,49]. Other work develops cryptographic protocols for achieving the same goal [39,44]. Shi et al. [51] propose a cryptosystem where an *aggregator* can compute the sum of multiple energy values from their ciphertexts, without access to the individual energy values because they have been encrypted under different keys. Rajagopalan et al. [43] propose a framework to quantify the privacy and usefulness of energy data and propose a model to control the tradeoff between them. This motivates us to build a system which employs this work for privacy-preserving application development for energy data.

System architectures: Previous work has focussed on energy data management for commercial buildings and office spaces, while aiming to achieve extensibility, scalability, and/or performance [11, 19, 25, 48]. Such systems are designed for users with expertise in the understanding of energy data (e.g., people specializing in building operations or energy managers). Residential consumers are unlikely to possess such levels of expertise, thus necessitating privacy-preserving third party applications (e.g., energy data analytics). To our knowledge our consumer-centric approach has been overlooked by existing work. Secondly, existing systems do not allow fine-grained access control over data streams, which is essential for privacy-preserving sharing of data.

Many other consumer-centric solutions [24,28,40,42,52,57] target various forms of personal data (e.g., healthcare, energy, mobile sensors, photos, videos). They provide data consolidation and ownership by aggregating data, but require exposing data to third parties thereby putting privacy at risk. Other work addresses data transformation before releasing it to third parties [20, 33, 34], consequently gaining privacy at the cost of inhibiting applications that require access to raw data. Càceres et al. [23,50] found that consumers' interests are best served by hosting their data on *virtual individual servers* in the cloud. We extend this approach to enable the in-depth analysis of consumer data such as time-series consumption data, by third party applications while preserving the privacy of the consumer.

3. GOALS AND REQUIREMENTS

Our main goal is to design a system that allows consumers to aggregate their data from multiple sources, control how that data is accessed and shared, and to allow them to quickly and easily access that data from any device, at anytime, from anywhere. These top-level goals translate into the following subgoals.

Consolidation: To allow a single view into multiple data streams and cross-correlation between different time series, the system should automatically consolidate energy usage data from multiple sources.

Durability: To allow analysis of usage history, a consumer's energy data should be always available, irrespective of its time of origin.

Portability: To prevent lock-in to a single provider, data and computation should be portable to different cloud providers.

Privacy: To preserve privacy, the system should allow a consumer to determine which other entities can access the data and at what level of granularity.

Flexibility: The system should allow consumers a free choice of analytic algorithms.

Integrity: The system should ensure that a consumer's energy data has not been tampered with by a third party.

Scalability: The system should scale to large numbers of consumers and large quantities of time series data.

Extensibility: It should be possible to add more data sources and analytic algorithms to the system.

Good Performance: Data analysis times and access latencies should be minimized.

Universal Access: Consumers should be able to get real-time access to their data on their Internet-enabled mobile devices.

4. DESIGN RATIONALE

We now describe the high-level rationale for our design by considering and eliminating several alternative approaches. The essential elements of any system that stores and analyses energy usage data are a data store, denoted D, and an application runtime that is the locus of execution of analytic algorithms, denoted AR.

The simplest possible system is one where the consumer stores data in their own home and uses a home-based computer for running data analytics. This case is shown as Case I in Figure 1. This solution provides portability, privacy, flexibility, and a certain amount of scalability, and extensibility. However, it requires consumers to be responsible for data collection and consolidation, and ensuring data durability (unfortunately very few consumers routinely backup their data). It also assumes that users' home computers are powerful enough to run sophisticated analytic algorithms over large data sets, which may not necessarily be the case, especially with the increasing proliferation of tablet devices. Moreover, if the home computer is behind a firewall, the solution does not provide good performance or universal access. For these reasons we do not believe this simple solution meets our design goals and the desires of consumers.

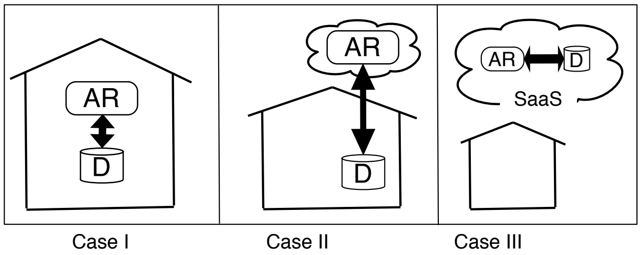


Figure 1: Solutions for energy data management. D: energy consumption data, AR: application runtime.

Consumers can avoid placing the computational burden on their home machines while, to some extent, preserving data privacy by storing data locally and sending data to analytic algorithms running in the cloud. This is shown as Case II in Figure 1. To preserve data privacy, data may be encrypted in a way that allows operations on ciphertext [51], or randomized values may be added to the data (i.e., dithering) to obfuscate details of consumer behaviour. This approach, however, allows a limited types of data computations and suffers from many of the same problems as the prior solution: the need for consumers to manage consolidation and durability, and the potentially poor performance.

Yet another approach would be to place both the data and the application runtime in the cloud using the "Software as a Service" (SaaS) approach. This is shown as Case III in Figure 1. Here, the SaaS provider would provide data consolidation and durability, freeing the consumer from these responsibilities. This approach, typified by the Microsoft Hohm [8] and Google Powermeter [5] approaches (both defunct), fails to provide privacy, extensibility, and flexibility, but does provide good performance and universal access.

Learning from the pros and cons of these three solutions, our goal is to provide a design that supports privacy, flexibility, and extensibility of data storage in the home combined with the consolidation, durability, good performance and universal access that can be obtained from a cloud-based solution. Specifically, we propose that a consumer shall have access to a "virtual home" or ***VHome*** that provides both data storage and an application runtime. Critically, the data access policies for data stores in the VHome are controlled not by the cloud provider, who would only be providing "Infrastructure as a Service" (IaaS), but by the consumer. As we demonstrate in Section 7 this solution meets all the design goals presented in Section 3.

By keeping the data and application runtime resident in the cloud, our solution allows VHome providers to support data consolidation and durability. Cloud-based data storage also allows low-latency universal access to the data, and relieves consumers of data consolidation and warehousing tasks. However, because consumers own their VHomes, they do not lose privacy or flexibility. We have engineered our solution for scalability, extensibility, and portability, as discussed later in the paper, thus meeting all of the design goals.

In the remainder of the paper, we present the details of our design, describe some applications we have implemented and evaluate whether or not our approach is successful in meeting our goals.

5. ARCHITECTURE

This section describes the architecture of our system. Details of our prototype implementation can be found in Section 6.

Figure 2 shows an overview of our system. It has four main components, from left to right, (a) the home-resident *gateway* (labelled Gateway), (b) the *virtual home* (labelled VHome) hosted by an SaaS provider in an IaaS cloud, (c) *cloud-based applications* (labelled CBA) also hosted in the cloud by other SaaS providers, and (d) *User interfaces* (labelled Remote UIs) for access to the gateway and the VHome from Internet-based devices. We now discuss each component.

5.1 Gateway

The gateway is a home-resident and consumer-controlled architectural element that provides two main services. First, it collects home energy production and usage data and uploads it over a secure connection to the cloud-based virtual home. Second, it provides an interface to allow the home owner to control devices in the home from Internet-connected devices.

The gateway interacts with smart appliances and monitors that are already network-capable and also, using add-on hardware such as Internet-controlled power strips, with legacy devices. Communication typically uses one or more types of channels such as USB, Zigbee, Ethernet, WiFi, RPL [56], or ZWave [17]. Usage data is uploaded from the home to the virtual home over a secure communication channel. This assures data durability and relieves the consumer from the need for data warehousing.

The gateway authenticates remote users and accepts control commands from them. These control commands either configure the gateway, request data uploads, or request that actions be taken by appliances and devices. A gateway may

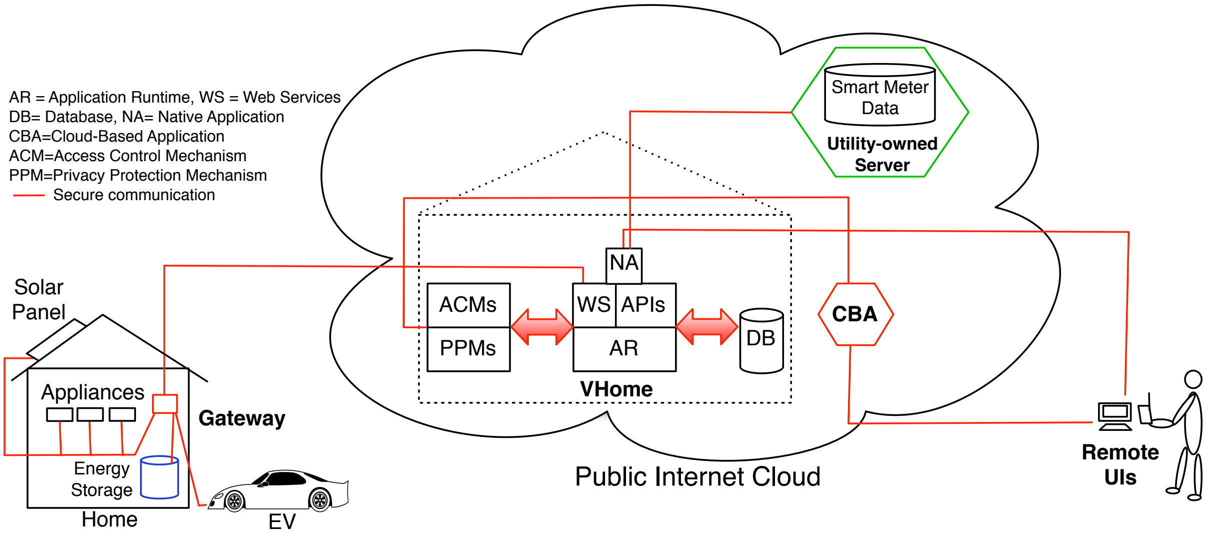

AR = Application Runtime, WS = Web Services
DB= Database, NA= Native Application
CBA=Cloud-Based Application
ACM=Access Control Mechanism
PPM=Privacy Protection Mechanism
— Secure communication
Smart Meter Data
Utility-owned Server
NA
ACMs
PPMs
WS
APIs
AR
DB
VHome
CBA
Solar Panel
Appliances
Gateway
Energy Storage
Home
EV
Public Internet Cloud
Remote UIs

Figure 2: System Overview.

be a dedicated, networked hardware device, or an integrated part of other home services' hardware such as a cable modem or set-top box, or it may be software deployed on a household computer.

5.2 VHome

A virtual home or VHome is a virtualized execution environment hosted in a cloud-based server that provides three services: (a) storage for home energy use data, (b) an application runtime for executing applications that analyse this data, and (c) trusted web-based services for interaction with the gateway, other cloud-based services, and user devices (described in more detail below). A VHome is owned by the consumer and hosted by a VHome SaaS provider in an IaaS cloud [1]. We describe the participation incentives for the consumer and these providers in Section 8.

A VHome is built from a virtual execution environment (VEE) provided by an IaaS provider. This could be a virtual machine [21] or a virtual container (private server) [53]. In Figure 2, the virtual execution environment is shown as a dotted home. Within the VEE, AR denotes the application runtime (such as a Java Virtual Machine) and DB denotes data stored in a database.

We envision that data stored in the database can be accessed by two types of applications. *Native* applications run on the VHome AR, and are certified to be "safe" using an approach described in more detail in Section 5.3. In contrast, *cloud-based applications* (denoted as CBA in Figure 2) pull energy data out of the VHome, which may violate consumer privacy. Therefore, access by a CBA to private data is mediated by privacy protection mechanisms (PPMs) that preprocess data before it is transferred out of the VHome. PPMs can implement privacy models such as differential privacy and k-anonymity [55] by employing mechanism such as obfuscation, noise addition, and homomorphic encryption [29]. An example of a PPM is to add random noise values to a meter reading, with the amplitude of the noise decreasing with reading granularity, so that monthly readings may have little or no added noise, but per-second readings would have large amounts of added noise. Access Control Mechanisms (ACMs) additionally allow consumers to restrict and revoke CBA access to the APIs by scope and duration. For example, an ACM may allow a CBA to access only hourly meter readings and only from a specified day of the year. Moreover, this access may expire after 15 minutes.

In addition to the native and cloud-based applications, our system contains special-purpose trusted applications we call *Web Services (WS)*. As a trusted component of the VHome they have free access to the APIs and hence to the energy data. They perform three tasks. First, they periodically accept data (typically, but not necessarily, bulk data) uploaded from the gateway and store it in the database. Second, they fetch real-time data from the gateway when requested by the consumer. This allows bulk data to be transferred from the home to the VHome once a day, yet provide real-time data access when necessary. Third, they provide a control interface to the consumer for various administrative tasks, such as downloading and running native VHome applications, configuring ACMs, configuring Privacy Protection Mechanisms (PPMs), requesting VHome software updates, the migration or discarding of data, and configuring gateway actions.

5.3 Applications

We now discuss native and cloud-based applications in more detail. Note that the main difference between native and cloud-based applications is that native applications execute in a tightly-controlled runtime environment. Moreover, their bytecode is available for analysis. This allows the system to eliminate the privacy leakage that is possible due to these applications. In contrast, cloud-based applications cannot be tightly controlled. Therefore, the only way to preserve privacy when giving data to these applications is to modify the data itself, which we accomplish using the PPMs.

We envision that both classes of applications would be developed by third-party developers, much like those who participate in Apple's App Store. Developers would use standardized APIs, such as those described next, to access con-

[1] This separation could be used in many other domains such as data management in healthcare, or banking.

sumer data. Consumers would either download native applications to the VHome to execute within the VHome runtime or can use ACMs to give cloud-based applications access to their data (after processing by PPMs). Applications can have user interfaces (UIs) to enable their invocation from PCs, smart phones, or other Internet-enabled devices.

Native Applications (NAs): The leakage of private data from native applications can be restricted using one of the following approaches. In the first approach, a native application's executable is scanned to assure consumers that the application is incapable of network communication. Thus, the application cannot leak data out of the container, which guarantees privacy. In our preliminary implementation, we restrict native applications to be written in Java and not invoke native APIs. Our current thinking is that an application can be certified as safe if its bytecode does not use the Java.net API. This can be easily checked when a native application is submitted for inclusion into the application store.

The second approach is used for native applications that need to use the network API to access remote hosts, such as to scrape consumer energy data from a utility website. To deal with such applications, network communications from a native application are restricted to a specific IP address (or host name). For instance, a native application could be restricted to communicate only with the host name corresponding to a utility's web server. Moreover, read or write access from a native application to a database table can also be restricted. In the example data scraper application, it could be restricted to only write to the database, not read from it. As we show in Section 6, these restrictions on database access are easy to accomplish in our system. We can also restrict the set of web services APIs that Native Applications can access. This further limits their ability to compromise privacy.

Certified native applications are suitable for data mining, analytics, visualization, appliance control and home automation. Native applications can also obtain consumers' energy data from utilities and store it in the VHome DB, making them ideal for data consolidation, (e.g., maintaining copies of consumers' smart meter data recorded by their utility companies).

Cloud-Based Applications (CBAs): Unlike native applications, cloud-based applications are hosted using third parties' hosting services. The main purpose of a cloud-based application is to allow sharing and comparison of energy data between different VHomes. ACMs provide fine-grained access control over data (e.g., time series) which means VHome owners chose which part(s) of her data are shared with a CBA, and when is it shared (e.g., periodically). The challenge here lies in preserving privacy while allowing meaningful computations and comparisons. While certified native applications can be given access to raw data, data given to a CBA must be pre-processed using techniques that ensure that privacy is preserved. Examples of such pre-processing include obfuscation, noise addition and homomorphic encryption [29]. These actions are implemented by and configured using the PPMs. Similar to NAs, CBAs can be published on the application store, and VHome owners provide CBAs with their VHome URL and explicit authorization to read all or parts of their data.

5.4 User interfaces

The gateway, a VHome's web services (WS), and cloud-based applications all allow user interaction. These interactions are mediated using user interfaces implemented on a user device, such as a web browser, a mobile application, or other mediums like e-mail or SMS. User interfaces simplify the management and use of VHomes and applications using graphical interfaces. Examples of such user interfaces are those used to download native applications to a VHome, to configure the permissions granted to a CBA by a consumer, and to control appliances in the home from a mobile device.

6. IMPLEMENTATION DETAILS

This section presents the details of a prototype implementation of our system.

6.1 Gateway

We implement a software-based gateway using the Microsoft HomeOS [26] platform. HomeOS is a .NET based platform designed to provide centralized control of devices in the home (such as light switches, thermostats, cameras, and televisions). It provides developers with homogeneous abstractions to orchestrate such devices. We use these features for monitoring and controlling appliances and to enable the uploading of data to the VHome.

Figure 3 provides an overview of HomeOS [27]. It is comprised of software modules called *drivers* that communicate with devices and allows higher level modules (such as applications) to actuate the devices. Additionally, a *platform* module manages and coordinates all other modules. In our gateway we extend HomeOS by implementing some additional modules, described next.

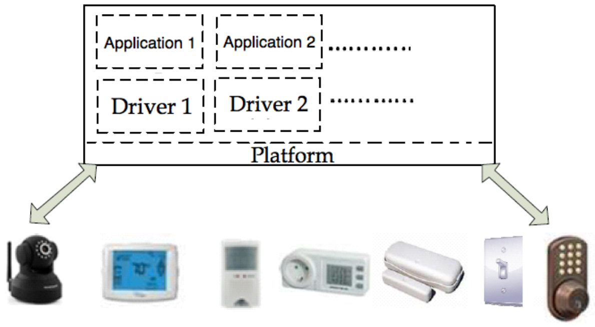


Figure 3: Overview of the HomeOS Platform.

Driver Modules

Each driver module monitors and controls an individual appliance using a sensor. We implement driver modules for the Aeon appliance sensor [1] and the CC Envi [4] power and temperature sensor. The Aeon sensor is installed in-series with an appliance and communicates to the gateway over Z-Wave [17]. The module is invoked by the coordinator module (described later) for polling data or controlling the sensor, and transmits the respective Z-Wave frames to the desired sensor. The Envi sensor measures the active power from a home every 6 seconds using a Hall-effect [7] transducer that is clipped around the split-phase wires at the home's main electricity supply. Measurements are transmitted wirelessly to the Envi console which is connected via a USB cable to the gateway machine (an inexpensive netbook in our prototype). The netbook stores the data on disk and transmits it periodically (e.g., once per day) to a VHome. Figure 4 shows the netbook running the gateway software, along with the Envi console.

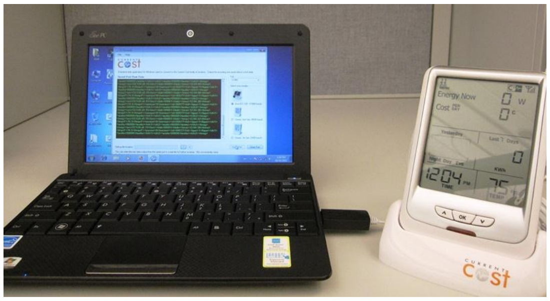


Figure 4: Gateway and Envi console.

Communication Module
This module provides communication between the VHome and the gateway. We considered several alternatives for communication including TCP, HTTP, and SSH. In the end, we decided to use XMPP [54], the protocol underlying the Jabber chat client, as our transport protocol because it uses a simple RPC mechanism that is secure, extensible, and provides real-time communication. Most importantly, XMPP ensures seamless communication from the VHome to the gateway despite the presence of NAT devices and firewalls at the home gateway.

Coordinator Module
This module records and processes energy data generated by the sensors' driver modules and caches it temporarily on the gateway. Periodic data uploads are received by the VHome's web services and are not sensitive to intermittent losses of connectivity. As a result, transient losses in network connectivity, lasting less than a few days, are easily tolerated. To facilitate coordination of sensor data and control between the gateway and the VHome, each sensor is assigned a *class ID* and an *object ID*. Sensors of the same type are assigned with the same class ID, but distinct object IDs. This allows the VHome to identify each sensor using the {class ID, object ID} tuple. The coordinator module uses the communication module to listen for commands from the VHome to control sensors. For example, if Aeon sensors have the class ID 1, and the one interfaced with the electric heater has an object ID 2, then the VHome issues the following command using XML to order the gateway to turn it off.

```
<setStatus classID=1 objectID=2>
  <power>0.0</power>
</setStatus>
```

The gateway performs the action and responds with the sensor's new status as an acknowledgement. Similarly other actions (e.g., dimming lights, managing AC temperature setpoints) can be performed using the VHome.

6.2 VHome

Each component of our prototype VHome, (e.g., the APIs, Web Services (WS), Access Control Mechanisms (ACMs), Privacy Protection Mechanisms (PPMs), and Native Applications (NAs)), is implemented as a Java Web Application (or *webapp*) using the Java API for RESTful Web Services (i.e., JAX-RS framework) [22], and can be deployed in a Java Web Container. We use Apache Tomcat as the web container, and MySQL as the relational datastore which we instantiate in a virtual machine using the Amazon EC2 [2] cloud. Our choice of Java was calculated to ensure VHome portability across cloud providers (e.g., Windows Azure [15] and RootBSD [12]) and our implementation is configurable to use any relational cloud datastore (e.g., SQL Azure [16] and Amazon RDS [3]).

Similar to the organization used for sensors (described in Section 6.1), data is organized into classes where each class describes a unique type of data stream and has a unique class ID, a class name (e.g., heating), a descriptor (e.g., space heaters in the home), and a rating (e.g., 500 W). Data streams either emanate from the sensors at home or could be external (e.g., smart meter readings from utility's website). Particular streams of a class are identified as objects using a unique object ID within their class, and have an object name (e.g., master bedroom heater), a descriptor (e.g., installed on 01/01/2011, warranted until 01/01/2020) and a granularity (e.g., 60, indicating data is produced every 60 seconds).

Privacy Protection Mechanisms (PPMs) are implemented as webapps and access a data stream or streams via APIs. They can create new privacy-preserving streams, which can then be shared with cloud-based applications (CBAs). For instance, we implement *aggregation* as an example PPM where energy consumption time series data (e.g., produced every second) is aggregated to compute daily or weekly consumption values which are less revealing in nature.

The webapp instantiating the VHome APIs implements them as a set of TLS-Secure Representational State Transfer (REST) [45] URIs, which are then used by native and cloud-based applications to access data or control sensors and thus appliances. Table 1 provides a brief overview of the APIs' URIs, which native and cloud-based applications can invoke using HTTP GET or POST requests. Results are returned using JavaScript Object Notation (JSON). Applications can add or modify data streams subject to the Access Control Mechanisms (ACMs). Applications can potentially compute a hash of a data stream (e.g., MD5) and sign it (e.g., using user's private key) to ensure data integrity to some extent.

The ACM webapp regulates applications' access to all or a subset of APIs, configured using the WS. By default all APIs are regulated and therefore, require a valid access token to return results. A VHome owner could decide to not regulate the *ListAllClasses* API, which in turn could result in a privacy breach. The ACM webapp implements OAuth 2.0 [32], a token based authentication and authorization standard for securing API access. It uses the VHome DB to store data concerning access controls (e.g., tokens, access lists, and more) which is only accessible to the ACM webapp.

Figure 5 illustrates this access process for a cloud-based application (CBA) hosted as a web portal. To access any API, the CBA is first required to obtain a *one time authorization grant* from the ACM webapp by providing its identity (identifier, name, or host-URL) and a list of APIs that it requires access to and the parameters to the APIs. For instance, a CBA requiring access to the bedroom space heater consumption data (e.g., with class ID 1 and object ID 2) from January–March 2012 would request access to the data stream API as:

Function (regulated by default)	**Description**
ListAllClasses	Returns all attributes of all classes of data in the VHome DB.
ListClass/param/value *param*: class ID, class name or rating.	Returns all attributes of class with given parameter values.
ListObject/param1/value1/param2/value2 *param1*: class ID, class name or rating. *param2*: object ID, object name or granularity.	Returns all attributes of object with given parameter values.
AddClass/className/x/descriptor/y/rating/z	Adds class with given class name, descriptor, and rating.
AddObject/classID/x/objectName/y/descriptor/z/granularity/w	Adds object with given class ID, object name, descriptor and granularity.
AddStream/classID/x/objectID/y/	Adds time series data to the given data stream.
GetStream/classID/x/objectID/y/	Returns the complete time series data stream.
GetStream/classID/x/objectID/y/TS/t_1/t_2	Returns the complete time series (or TS) between timestamps t_1 and t_2.
GetStream/classID/x/objectID/y/Val/v_1/v_2	Returns the complete time series between data values (or Val) v_1 and v_2.
GetStatus/classID/x/objectID/y	Returns the current power consumption of device with given class ID and object ID.
SetStatus/classID/x/objectID/y/status/p	Sets the power consumption of device with given class ID and object ID to p.

Table 1: VHome API used by NAs and CBAs to access data.

```
https://<VHome URL>/GetStream/classID/1/
        objectID/2/TS/1325394000/1333252799
```

where TS indicates time series data, and 1325394000 and 1333252799 are the epoch timestamps at 01-01-2012 00:00:00 and 31-03-2012 23:59:59, respectively.

This allows restricting the scope of data access to certain data streams and/or certain segments of a stream's time series defined using timestamp and/or data values. The ACM webapp then redirects the user to the Web Services (WS) webapp so as to authenticate the user as the VHome owner. After authentication, the scope and nature of the requested access is described to the user, and her authorization for the access is requested. The WS implements this as a simple notification in a web-browser, which can be relayed to other remote UIs such as email, SMS, or mobile application notification. An example of such a notification from an application named "EXAMPLE" is:

The application named EXAMPLE is requesting
access to bedroom space heater data
for Jan 1 to Mar 31, 2012.
Allow or Deny ?

The CBA then has to exchange the one time authorization grant before it expires and obtain an *access token* and an (optional) *refresh token*. By using the access token, a CBA can use the required APIs until the token expires after which a new access token may be obtained using the refresh token. All tokens are valid for periods configured by the owner of the data. By matching the CBA's credentials (e.g., URL) to those registered while issuing the authorization grant the ACM validates each API access and prevents use of stolen access tokens. Further, if at any point the user decides to revoke (or pause) a CBA's access to data, she can simply revoke the access token and possibly the refresh token for that CBA. Our prototype implements the authorization grant, access token and refresh tokens in the form of randomized 128-bit MD5 [46] codes, where the webapp maintains a lookup table for storing their scope and expiry times. Avoiding the encapsulation of scope and duration within the token circumvents token processing overhead for each API access. The authorization grant and access token issuing endpoints are published as GET/POST URIs by the VHome, and use JSON for token and error-message exchanges with CBAs. Our prototype implementation assumes one user per VHome, thus managing multiple users is not currently addressed.

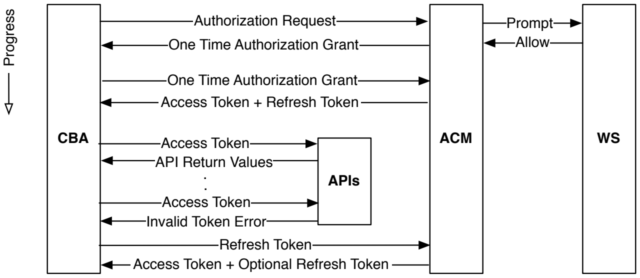


Figure 5: API access for a CBA.

The Web Services (WS) webapp is critical to a VHome. It enables a number of components and allows users to configure them. Firstly it coordinates periodic data uploads from the gateway over XMPP and transmits control commands to the gateway's coordinator module. Secondly it provides the consumer with a control portal to install native applications on the VHome. Native Applications, being JAVA web applications, are then profiled by the WS webapp for use of the JAVA.net interface and for the VHome APIs they require. The owner of the data (the consumer) can restrict the applications' ability read from and write to the database by disallowing or restricting the scope of the APIs. Likewise, consumers can configure ACM settings such as token formats and expiry periods, and chose which APIs it restricts. Similar to native applications, certified PPMs can be added to the VHome through this portal which can be run to create additional data streams. Lastly, the WS webapp allows users to purge native applications or data, and revoke access tokens of any CBA they want.

Our prototype VHome implementation is open source and can be found at `https://vhome.codeplex.com` while the gateway implementation using HomeOS is hosted at `http://homeos.codeplex.com/`.

6.3 Applications

We have created a sample application store as a web portal where users browse for native applications. It transfers the desired native applications' executables to the VHome' WS which installs them on the VHome, and can then be accessed using remote UIs. We now describe a few applications that we have built using our system which, without our architecture, cannot be implemented in a data privacy preserving form.

Data Scraper
This application obtains consumers' smart meter data from the utility provider and stores it in the VHome DB. It is implemented using the JAVA DOM interface as a VHome native application. Our prototype application is downloaded from the sample application store, to run on the VHome, where it scrapes data from the utility company's web portal and stores it in the VHome DB. When first using this application it obtains, from the consumer, their identification and password used to access the utility company's portal. The utility company in our prototype is Waterloo North Hydro [14]. The application also allows consumers to set automated periodic data scraping actions to ensure that data is obtained before it is discarded by the utility company's portal (i.e., after three months) and the consumer is relieved of manually retrieving the data. The application allows the data to be retained by the consumer even after it is no longer available on the utility company's portal. The data is stored as a data stream in the *Smart Meter* class which can then be accessed by other applications through APIs. Access to more than a year's worth of data can provide excellent opportunities for data analytics because in many climates seasonal changes must be accounted for when examining consumption histories. This application demonstrates how our architecture allows consumers to take ownership of their data, thus meeting the goal of *data ownership*.

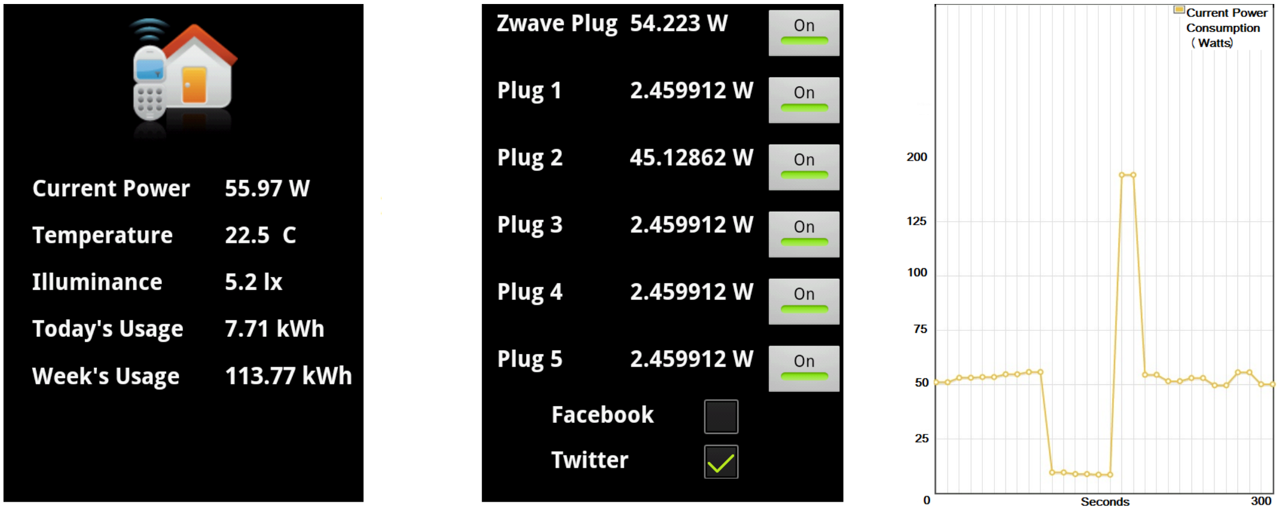
Interactive Monitoring and Control
We have implemented a VHome native application that interfaces with the VHome web services to allow the consumers to use a web-browser to monitor and control home appliances in real-time. Native applications have no network access and can only be viewed by invoking the trusted VHome webapp container. We implement an Android smartphone application that invokes the VHome native application via the VHome webapp container and provides a smartphone application interface. This means consumers can use VHome native applications via web-browsers or with applications installed on their mobile devices (e.g., smartphones, or tablets). Figure 6 shows snapshots of different panels in the Android smartphone application. Screenshot-1 (on the left) shows the home monitor, which allows consumers to view current conditions of the home as reported by the CC Envi console. In addition, the consumption data stored at the VHome is used to compute and display the day's and week's consumption. Screenshot-2 (in the center) shows the control panel which allows users to turn on or turn off different appliances connected to Aeon ZWave sensors and displays their current consumption. Further, it allows users to share the amount of energy they conserve by turning off appliances, on social networks such as Facebook and Twitter and to compete with their friends. This applications permits consumers to define events for which they wish to receive notifications. For instance, consumers can set a threshold for energy consumption and "abnormal consumption" notifications are issued if it is exceeded. Screenshot-3 (on the right) shows a past trend of aggregate energy consumption measured using the CC Envi sensor. This trend data can be used by consumers to better understand abnormal consumption notifications. Our prototype implements SMS, E-Mail and mobile application notifications. Since the VHome is cloud resident, energy data can be processed in the cloud and viewed using any Internet-enabled device, with relatively low latency.

Energy Data Analytics
In many parts of the world the price of electricity depends on the time of the consumption. In Ontario, a day is divided into peak, mid-peak, and off-peak hours, each with different rates [9]. We implement a VHome native application that processes a home's electricity consumption measured using the CC Envi sensor to determine how much energy is consumed during different hours of the day, its respective costs under the pricing scheme, and the total cost. It uses smart meter data obtained and stored by the data scraper application to verify a consumer's utility bill. Such simple analytics also provides consumers with meaningful insight into their hourly and daily consumption patterns, warns them of potential errors in their utility bills and can help them to time shift non time-critical consumption. This shows how our architecture meets the goal of *data analytics*.

Abnormal Energy Consumption Detection
We implement a VHome NA which informs consumers about abnormalities in their energy consumption. For instance, consider a scenario where residents forget to turn off their oven while they are away. Using the VHome APIs the application periodically obtains the energy consumption values from the gateway, measured by the CC Envi sensor. It then compares the values to a predicted value computed using an Auto-Regressive Moving Average model. If the measured value is higher than the predicted value by a threshold (e.g., 1 kW) then the application sends the consumer a notification message via e-mail, SMS, or the Android smartphone application. The consumer can then either use the Android application (described above) or reply to the email, SMS to take appropriate action. We defer the use of more complex abnormality detection techniques to future work.

7. EVALUATION

In this section, we compare our architecture to the different existing and proposed systems that can be used to store and analyse energy data. Table 2 compares these solutions by denoting which of the requirements of a consumer-centric solution they meet (i.e., the goals from Section 3). Commercial software solutions–Google Powermeter [5] and Microsoft Hohm [8] being centralized web services are scal-



Current Power	55.97 W
Temperature	22.5 C
Illuminance	5.2 lx
Today's Usage	7.71 kWh
Week's Usage	113.77 kWh

Zwave Plug	54.223 W	On
Plug 1	2.459912 W	On
Plug 2	45.12862 W	On
Plug 3	2.459912 W	On
Plug 4	2.459912 W	On
Plug 5	2.459912 W	On

Facebook □

Twitter ☑

Figure 6: Three screenshots of the Android smartphone application.

able but provide a fixed set of analytics with no data consolidation or privacy. Both services are now defunct leaving consumers with no data access. Utilities' web portals act similarly and share/discard data at their discretion but can ensure integrity of only smart meter data. Opower [10] provides consumers with some data analytics on their monthly utility bills, but they provide no real-time access to data or choice of analytics. The Greenbutton [6] initiative has standardized energy data formats, so that consumers can access their data and analyse it themselves (denoted "Greenbutton (Self)"). This allows consumers to choose analytics tools, run integrity checks on data, and provides data privacy, but it burdens them with data maintenance. Alternatively, consumers can delegate the data retrieval and maintenance to third parties (denoted "Greenbutton (Third Party)"). Third parties can manage, analyse and host consumers' data, but such unconditional access to raw data provides little data privacy.

Our architecture meets the requirements for a consumer-centric, privacy preserving, architecture for energy data analysis as explained below.

Consolidation: Native applications allow data to be read from any data source and stored in the VHome database. This allows consumers to easily consolidate their data from multiple sources by using one native application per data source.

Durability: The cloud-based storage of data allows for data durability. Instead of relying on a single computer in the consumer's home to store data, and relying on the consumer to remember to back up that data and to ensure that they have off-site backups, data is stored in the cloud. The data on cloud-based storage is replicated, backed-up, and maintained by professionals, guaranteeing durability.

Portability: Our solution has been designed to be portable across a variety of cloud-based providers and databases. This ensures application and data portability, as discussed in detail in Section 6.

Privacy: Our system provides data privacy in several ways. First, data within a VHome is not accessible by entities outside the VHome, eliminating most types of privacy violations. Privacy leakage from native applications is prevented by certifying native applications, by checking Java byte code submitted to the application store to ensure that they are either not using network APIs, or when they need to, are only communicating with the specified hosts. The details of this process are described in Section 6.2. Privacy leakage from cloud-based applications is mitigated, to some extent, by data encryption or obfuscation by privacy preserving mechanisms. Of course, this protection of data privacy assumes that Iaas and VHome SaaS providers are trusted.

Flexibility: Our system allows consumers to download and install analytic algorithms of their choice. They can also send data to be processed by any cloud-based application by giving them time-limited, scope-limited access to their data.

Integrity: Native applications allow integrity for data stored directly in the VHome (e.g., via gateway) but cannot ensure integrity for data imported from external sources (e.g., smart meter data from utility company servers).

Scalability: Cloud-based servers allow massive scaling of both data set sizes and computation, unlike the use of home-based computers.

Extensibility: The native application store allows a consumer to extend their VHome with new analytic algorithms. Consumers can also send their data to any cloud-based application for analysis.

Good Performance: The use of a cloud to store data minimizes data access latencies by avoiding the use of the typically lower-bandwidth home access link. In addition it provides access to server systems with more memory and processing power than may be available on many consumer's home machines.

Universal Access: Cloud-resident data allows consumers to get real-time access to their data on their Internet-enabled mobile devices.

Thus, our system meets all of our design criteria, while our prototype implementation demonstrates the feasibility

Solutions Goals	Hohm [8], Powermeter [5]	Utility web portals [13, 14]	Opower [10]	Greenbutton [6] (Self)	Greenbutton [6] (Third Party)	VHome
Consolidation				✓	✓	✓
Durability						✓
Portability				✓	✓	✓
Privacy				✓		✓
Flexibility				✓	✓	✓
Integrity		*		✓		*
Scalability	✓	✓	✓		✓	✓
Extensibility				✓	✓	✓
Performance	✓	✓			✓	✓
Universal access	✓	✓			✓	✓

Table 2: Comparative analysis with existing solutions, * denotes a partial solution.

of such a system using existing hardware, software and cloud infrastructure.

8. DISCUSSION

In some sense, ensuring that meter data remains private is moot, because utility companies already collect this data and share it with whomever they choose (e.g., Google PowerMeter and Microsoft Hohm) without seeking customers' permission. This situation, however, is likely to change in two ways in the future.

First, we anticipate that many jurisdictions, following the lead set by the province of Ontario (in Canada), will place severe restrictions on the sharing of meter data, thereby freezing innovation in data analytics and customized recommendations. Although this is being countered by proposals such as the Green Button initiative [6], which release usage data back to consumers, we believe that consumers are just not capable of doing their own data analytics, and are loath to share this data with third parties due to privacy concerns. Second, besides meter data, we anticipate that future consumers will generate many other equally private data streams including health-monitoring data. Our architecture balances privacy and innovation for applications that analyse these other data streams.

The entities that participate in our system are the utility companyies that collect smart meter data, consumers, IaaS providers, VHome SaaS providers, and application developers. We believe that each entity has an incentive to participate in the system.

Utilities: Utilities are under great pressure from legislatures to release usage data, as evidenced by the Green Button initiative. They also benefit from energy conservation measures, in that these reduce their need for costly generation capacity upgrades.

Consumers: Consumers are increasingly aware of the costs of the world's rampant energy consumption. In some cases consumers will be motivated to better understand and reduce their consumption by trying to improve the health of the planet and in other cases they will be motivated by trying to reduce their utility bills. They currently lack the infrastructure and tools to understand how to achieve reductions without giving up their privacy.

IaaS providers: IaaS providers are paid for their services, so they have a monetary incentive for participation.

VHome SaaS providers: We believe that VHome SaaS providers can be compensated for their efforts in two ways. First, some consumers may wish to pay for their own VHomes, so that they can maintain an archive of past usage and get recommendations for intelligent energy use. This is similar to those consumers who pay a monthly fee for services such as DropBox. Second, vendors of "green" energy-efficient products could subsidize the cost of VHomes, because recommendations for the use of their products, such as energy-efficient air conditioners, washing machines and LED lights, translate to increased sales.

Application developers: Application developers have the same incentives in our architecture as with the Apple App store: a mass audience for their work, so that a well-developed application can make its developer a lot of money. Certain applications may also be commissioned by equipment vendors, as discussed above.

In hindsight, our approach may appear to be obvious, merging the cloud with sensor data streams, an approach already implemented by systems such as Pachube [11]. However, there are three aspects of our work that are not obvious. First, we show how to use virtualized execution environments, in conjunction with an object-level framework, to provide practical solutions for the seemingly conflicting requirements of ensuring data privacy while fostering application development. Second, our approach enables the development of an ecosystem of energy management applications in much the same way as Apple's App Store provides an ecosystem for the development of iPhone and iPad applications. Third, our approach is diametrically opposed to the utility-centric view that is widely prevalent in the Smart Grid community. Instead of designing an architecture that caters to the needs of utilities, our approach places control firmly in the hands of consumers.

9. LIMITATIONS

Our prototype implementation demonstrates that it is possible to provide an infrastructure that enables the analysis of consumer energy consumption data while perserving their privacy but it suffers from certain limitations. Our current implementation is limited to dealing with time series and does not support other potential forms of energy data. Because it provides consumers with much greater control over their data, the consumer is faced with many decisions. Such cognitive overload may be eased by learning users' perceptions from user studies to help make decision-making simple and intuitive. Although fully realizable in our architecture, our current prototype implementation does not include mechanisms for ensuring integrity of data streams (e.g., by maintaing signed hashes). Our architecture also requires a trusted certification mechanism to certify applications and requires VHome SaaS and IaaS cloud providers to be non mallicious. Lastly, to gain data privacy, the consumer may have to bear the cost of hosting the VHome in the cloud and purchasing analytic applications.

10. CONCLUSIONS

We describe how cloud hosting services can be leveraged to ensure that consumers retain ownership and fine-grained control over their energy consumption data while enabling third party applications to analyse that data in a privacy preserving fashion. In addition, our cloud-based architecture provides applications with low-latency data access, long-term, durable storage, and access to the significant computational and storage resources needed to process growing volumes of energy data being collected. We believe our architecture is essential for the development, management and automation of applications that provide intelligent, privacy-preserving, energy data analytics. We defer the study of the scalability and performance of our prototype implementation for future work.

11. ACKNOWLEDGMENTS

The authors wish to thank the European Institute of Network Sciences (EINS), Natural Sciences and Engineering Research Council of Canada (NSERC), and MITACS for their financial support, and Ratul Mahajan from Microsoft Research for sharing HomeOS with us.

12. REFERENCES

[1] Aeon Labs Smart Energy Switch. www.aeon-labs.com.

[2] Amazon Elastic Compute Cloud (EC2). www.aws.amazon.com/ec2.

[3] Amazon Relational Database Service (RDS). www.aws.amazon.com/rds.

[4] Current Cost Envi CC-128. www.currentcost.com.

[5] Google Powermeter. www.google.com/powermeter.

[6] Green Button Initiative. www.greenbuttondata.org.

[7] Hall effect. http://en.wikipedia.org/wiki/Hall_effect.

[8] Microsoft Hohm. www.microsoft-hohm.com.

[9] Ontario time-of-use pricing. www.ontario-hydro.com.

[10] OPower Inc. www.opower.com.

[11] Pachube-Cosm Ltd. www.cosm.com.

[12] RootBSD Cloud Provider. www.rootbsd.net.

[13] San Diego Gas and Electric. www.sdge.com.

[14] Waterloo North Hydro Corp. www.wnhwebpresentment.com/app/.

[15] Windows Azure. www.windowsazure.com/.

[16] Windows SQL Azure. www.windowsazure.com/en-us/home/features/data-management/.

[17] Z-Wave Alliance. www.z-wavealliance.org.

[18] G. Ács and C. Castelluccia. I have a DREAM!: differentially private smart metering. In *Proc. of IH*, 2011.

[19] Y. Agarwal, R. Gupta, D. Komaki, and T. Weng. Buildingdepot: an extensible and distributed architecture for building data storage, access and sharing. In *Proc. of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys, 2012.

[20] P. Arjunan, N. Batra, H. Choi, A. Singh, P. Singh, and M. B. Srivastava. SensorAct: a privacy and security aware federated middleware for building management. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '12, pages 80–87, New York, NY, USA, 2012. ACM.

[21] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 2003.

[22] B. Burke. *RESTful Java with Jax-RS*. O'Reilly Media, 2009.

[23] R. Cáceres, L. Cox, H. Lim, A. Shakimov, and A. Varshavsky. Virtual individual servers as privacy-preserving proxies for mobile devices. In *Proc. of ACM MobiHeld, 2009.*

[24] H. Choi, S. Chakraborty, Z. M. Charbiwala, and M. B. Srivastava. Sensorsafe: a framework for privacy-preserving management of personal sensory information. In *Proceedings of the 8th VLDB international conference on Secure data management*, SDM'11, pages 85–100, Berlin, Heidelberg, 2011. Springer-Verlag.

[25] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, and D. Culler. sMAP: a simple measurement and actuation profile for physical information. In *Proc. of ACM SenSys*, 2010.

[26] C. Dixon, R. Mahajan, S. Agarwal, A. Brush, B. Lee, S. Saroiu, and V. Bahl. An operating system for the home. *Proc. NSDI*, 2012.

[27] C. Dixon, R. Mahajan, S. Agarwal, A. J. Brush, B. Lee, S. Saroiu, and V. Bahl. The home needs an operating system (and an app store). In *Proc. of HOTNETS*, 2010.

[28] C. Elsmore, A. Madhavapeddy, I. Leslie, and A. Chaudhry. Confidential carbon commuting: exploring a privacy-sensitive architecture for incentivising 'greener' commuting. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, 2012.

[29] F. D. Garcia and B. Jacobs. Privacy-friendly energy-metering via homomorphic encryption. In *Proc. of the 6th International Conference on Security and Trust Management*, 2010.

[30] U. Greveler, B. Justus, and D. Loehr. Multimedia Content Identification Through Smart Meter Power Usage Profiles. In *Proc. of CPDP*, 2012.

[31] H. Haddadi, R. Mortier, S. Hand, I. Brown, E. Yoneki, J. Crowcroft, and D. McAuley. Privacy analytics. *SIGCOMM Comput. Commun. Rev.*, Apr. 2012.

[32] E. Hammer-Lahav, D. Recordon, and D. Hardt. The OAuth 2.0 authorization protocol. *draft-ietf-oauth-v2-18*, 8, 2011.

[33] J. Kannan, P. Maniatis, and B.-G. Chun. A Data Capsule Framework For Web Services: Providing Flexible Data Access Control To Users. *CoRR*, 2010.

[34] J. Kannan, P. Maniatis, and B.-G. Chun. Secure data preservers forweb services. In *Proceedings of the 2nd USENIX conference on Web application development*, WebApps'11, pages 3–3, Berkeley, CA, USA, 2011. USENIX Association.

[35] Y. Kim, E.-H. Ngai, and M. Srivastava. Cooperative state estimation for preserving privacy of user behaviors in smart grid. In *Proc. of SmartGridComm*, 2011.

[36] F. Li, B. Luo, and P. Liu. Secure Information Aggregation for Smart Grids Using Homomorphic Encryption. In *Proc. of SmartGridComm*, 2010.

[37] W. Liu, K. Liu, and D. Pearson. Consumer-centric smart grid. In *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*, pages 1–6. IEEE, 2011.

[38] D. McAuley, R. Mortier, and J. Goulding. The Dataware manifesto. In *Proc. of COMSNETS*, 2011.

[39] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *Proc. of ACM BuildSys*, 2010.

[40] R. Mortier, C. Greenhalgh, D. McAuley, A. Spence, A. Madhavapeddy, J. Crowcroft, and S. Hand. The personal container, or your life in bits. *Digital Futures*, 2010.

[41] R. Mortier, C. Greenhalgh, D. McAuley, A. Spence, A. Madhavapeddy, J. Crowcroft, and S. Hand. The Personal Container, or Your Life in Bits. *Digital Futures*, 2010.

[42] M. Mun, S. Hao, N. Mishra, K. Shilton, J. Burke, D. Estrin, M. Hansen, and R. Govindan. Personal data vaults: a locus of control for personal data streams. In *Proc. of ACM CoNext, 2010.*

[43] S. R. Rajagopalan, L. Sankar, S. Mohajer, and H. V. Poor. Smart Meter Privacy: A Utility-Privacy Framework. *CoRR*, 2011.

[44] A. Rial and G. Danezis. Privacy-preserving smart metering. In *Proc. of ACM WPES*, 2011.

[45] L. Richardson and S. Ruby. *RESTful web services.* O'Reilly Media, Incorporated, 2007.

[46] R. Rivest. The md5 message-digest algorithm. 1992.

[47] C. Rottondi, G. Vertical, and A. Capone. A security framework for smart metering with multiple data consumers. In *Proc. of IEEE INFOCOM 2012 Workshop: Green Networking and Smart Grid*, 2012.

[48] A. Rowe, M. E. Bergeés, G. Bhatia, E. Goldman, R. Rajkumar, J. H. Garrett, J. M. F. Moura, and L. Soibelman. Sensor andrew: large-scale campus-wide sensing and actuation. *IBM J. Res. Dev.*, Jan. 2011.

[49] S. Ruj, A. Nayak, and I. Stojmenovic. A Security Architecture for Data Aggregation and Access Control in Smart Grids. *CoRR*, 2011.

[50] A. Shakimov, H. Lim, R. Caceres, L. Cox, K. Li, D. Liu, and A. Varshavsky. Vis-a-Vis: Privacy-preserving online social networking via Virtual Individual Servers. In *Proc. of COMSNETS, 2011.*

[51] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-Preserving Aggregation of Time-Series Data. In *NDSS*, 2011.

[52] K. Shilton, J. Burke, D. Estrin, R. Govindan, M. Hansen, J. Kang, and M. Mun. Designing the personal data stream: Enabling participatory privacy in mobile personal sensing. TPRC, 2009.

[53] S. Soltesz, H. Pötzl, M. Fiuczynski, A. Bavier, and L. Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In *Proc. of EuroSys*, 2007.

[54] P. St-Andre. Extensible Messaging and Presence Protocol (XMPP). *IETF Network Working Group, RFC3920*, 2004.

[55] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2002.

[56] T. Winter and P. Thubert et al. RPL: Ipv6 Routing Protocol for Low power and Lossy Networks. Internet Draft, IETF, 2010.

[57] R. Wishart, D. Corapi, A. Madhavapeddy, and M. Sloman. Privacy butler: A personal privacy rights manager for online presence. In *Proc. of Workshop of Smart Environments, 8th IEEE PerCom*, 2010.