Customer-centric Energy Usage Data Management and Sharing in Smart Grid Systems

Gaurav Lahoti*
University of Illinois at Urbana-Champaign
Urbana, IL
lahoti2@illinois.edu

Daisuke Mashima, Wei-Peng Chen
Fujitsu Laboratories of America
Sunnyvale, CA
{dmashima, wchen}@us.fujitsu.com

ABSTRACT

A recent study revealed that fine-grained energy usage data could potentially leak sensitive information about an electricity customer. On the other hand, a number of online service providers utilizing such data have emerged and improved effectiveness of smart grid technologies (for instance, demand-response aggregators), and therefore sharing of data is getting popular. In this work, we propose a customer-centric framework to manage, store, and share energy usage data in a privacy-enhanced way. We present a mechanism to enable customers to flexibly control the amount of energy usage information disclosed while still allowing third-party service providers to be convinced of the authenticity of data. A prototype implementation using the widely used Green Button data model is presented and evaluated. We further discuss the design of a demand-response aggregation service on top of the proposed framework.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Data Sharing*

General Terms

Security, Verification

Keywords

Data privacy, Green Button, demand-response aggregation

1. INTRODUCTION

Broad penetration of smart meters and advanced metering infrastructure (AMI) has enabled bidirectional communication between utility companies and customers and collection of fine-grained energy consumption data. It benefits both the generation and distribution side (i.e., utility companies and independent service operators) and the demand side (i.e., electricity customers). For instance, a utility can use the corrected data to better predict peak demand to avoid service outages and improve the stability of the grid. Also, a utility can have more control, either directly or indirectly, over its customers' energy consumption pattern, for example by means of direct load control or demand response, which may result in lower operational cost for additional generation. On the other hand, customers can benefit by knowing and optimizing their energy consumption patterns to improve energy efficiency. In addition, home energy management systems (HEMS) or building energy management systems (BEMS) can use energy consumption data to automatically control appliances on the customer's premises. Other innovative new technologies that utilize such data and communication infrastructure are also emerging.

The landscape around data management and sharing in smart grid systems is getting more complicated. For instance, data analytics on customers' energy usage data is often outsourced by utility companies to third parties. Example is the partnership between PG&E, which is the largest utility company in California, and Opower, a software-as-a-service company located in Virginia which provides PG&E recommendation services [17]. Moreover, in order to facilitate large-scale demand-response services, recently a number of third-party service providers have begun offering the services of *Demand-Response aggregators (or DR aggregators)*. Representative examples of DR aggregators include EnerNOC [3] and ECS [1].

Such third-party services are beginning to play an important role in the smart grid, and customers' privacy is not the first priority under the current system model. For instance, EnerNOC [3] installs its own metering device at each customer's site so that it can obtain fine-grained meter reading data to facilitate its services. Thus the customer's energy usage data, which are often called *CEUD* (Customer-specific Energy Usage Data), combined with personally identifiable information like billing information, is collected by DR aggregators as well as by utility companies. While the latter have a justifiable reason, agreed to by customers, for collecting such data, the former may not. DR aggregators could potentially gather as much information as they want, regardless of whether it is needed for providing services. A similar concern could be raised if services are outsourced to a third party and customers do not have direct control over data sharing. Such issues not only violate the "minimal disclosure" principle, which is desired for privacy protection

*The work was done while the author was with Fujitsu Laboratories of America.

SEGS'13, November 8, 2013, Berlin, Germany.

http://dx.doi.org/10.1145/2516930.2516935.

in cyberspace [16], but also, given the possibility of unauthorized data usage and authorized or unauthorized sharing and unintentional information breach, increase the risk of jeopardizing customers' privacy, for example, by means of methods called Non-intrusive Load Monitoring (NILM) [20] or Non-intrusive Appliance Load Monitoring (NALM) [23]. Moreover, breach of sensitive data handled by third-party service providers, whether through hacking, malware, or disgruntled insiders, would be a serious concern. Therefore, from the customers' perspective, it is desirable to minimize the amount of data shared or disclosed to third parties while obtaining maximal benefit with respect to services. On the other hand, to provide services, third parties need to ensure that the data provided are the actual data as collected and supplied by the utility and have not been modified. Since a malicious customer may falsify its data to skew its personal usage history to gain illegitimate advantages, that is especially the case when accounting and billing are involved, such as in DR aggregation services.

In this work, we propose a customer-centric framework for energy usage data management that emphasizes customer privacy as well as data verifiability for third-party service providers. This paper is organized as follows. We first summarize the goals and contribution of this work in Section 2. In Section 3 we discuss the customer-centric energy usage data management framework, including cryptographic primitives, that we designed to meet the goals. Section 4 discusses the implementation of the framework using the widely used Green Button data model [6]. Limitations of the standard schema and enhancement of it are also discussed. In Section 5, we present one application of the proposed mechanism to provide customer-centric demand-response aggregation services. Section 6 discusses related work, and we conclude in Section 7 with a discussion of future work.

2. GOALS AND CONTRIBUTIONS

Just as in the e-healthcare domain [28], an argument about the ownership of data is taking place in the smart grid area, and we advocate the standpoint that each customer shares the ownership of his or her own energy consumption data and therefore should retain awareness and control over it even if data are collected and stored by utility companies. In this section, we list the requirements for a customer-centric system architecture for energy usage data management in a setting where third-party service providers are involved. Specifically the following goals must be met to minimize customers' privacy concerns.

- Energy usage data are accessible to the customer to whom the data are pertinent.
- Sharing of energy usage data is done at customers' discretion.
- Customers should be able to control the amount of personal energy usage information to be disclosed upon sharing.

At the same time, we need to ensure that the shared energy usage data is meaningful for third-party service providers. We believe that verifiability of data is key. In other words, if authenticity of data cannot be verified by a third party, that third party would hesitate to use those data for certain types of services, such as demand-response aggregation for which customers' receive monetary incentives according to their performance. Without such verifiability, the types of services that can be provided and the effectiveness of smart grid technologies would be significantly limited. Thus, the fourth goal is:

- The authenticity and integrity of energy usage data shared by customers are publicly verifiable while meeting the above privacy goals.

Our contributions towards meeting these goals include:

- A system design involving a customer-controlled repository that stores a customer's energy usage data and handles data-sharing in a privacy-preserving way.
- A Merkle Hash Tree based design for redactable, verifiable energy usage data.
- A prototype implementation of the proposed mechanism in an extended Green Button data model.
- An application of the proposed framework to realize a customer-centric demand-response aggregation service model.

3. APPROACH FOR CUSTOMER-CENTRIC ENERGY USAGE DATA MANAGEMENT

3.1 System Architecture

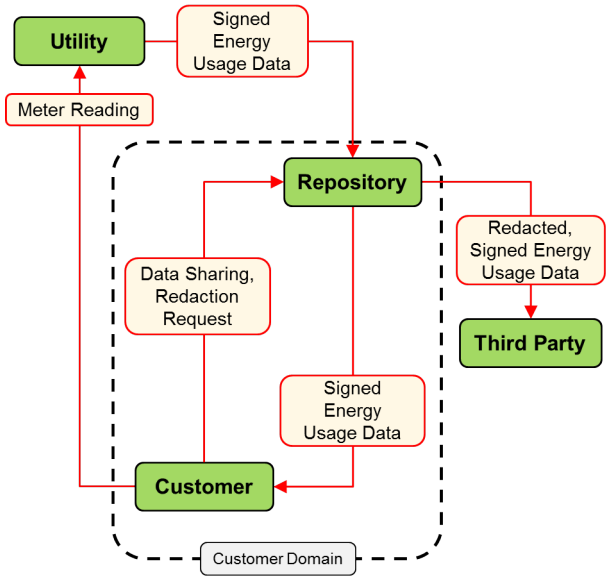

Figure 1: High-level View of Customer-centric Energy Usage Data Management and Sharing

This section describes the architecture of our system. Figure 1 presents the participating entities and illustrates the information flow among the entities. As can be seen in the figure, there are four participating entities in the system.

The central entity to enable customer-centricity is a data repository (*Repository* in the figure) controlled (and therefore trusted) by a customer (*Customer*). The repository can be a storage module hosted on a customer's own computer, or can be a server space hosted by a service provider

that the customer can trust. In the latter case, the concept of the repository is analogous to a Personal Healthcare Record (PHR) system in the e-healthcare domain [9]. Customers can download their energy usage data from a utility system (*Utility*) and store it on their repository, or the repository can periodically or automatically download the data from the utility on the customers' behalf. In either case, as explained below, the data are digitally signed by the utility. We use the term "utility" in this work, but it can be any entity that collects and manages customers' energy usage data, for instance independent system operators. Customers can access and browse the data stored on the repository whenever they want. Such access or control over the repository may be possible via dedicated client software or a Web browser. Additionally, customers can issue commands to the repository when they intend to share some data with a third party service provider. Here, the repository is responsible for generating a minimal-disclosure form of energy usage data based on the data originally provided by the utility. In other words, customers can redact part of the data that they think is irrelevant to the expected service or that they do not want to share with the third party. Such a process can also be automated through definition of data-sharing policy on customers' repository.

Data provided by the utility are originally collected by smart meters that are installed on customers' premises. We assume that energy consumption data are measured in an interval (for instance, 1 minute or 5 minutes) that is short enough that the resulting data set can be used for a variety of services, and is reported back to the utility periodically (for example, hourly or every 15 minutes). The bandwidth in the metering infrastructure is often limited, but reporting in 15 minutes intervals or longer is realistic, as demonstrated in [24]. Before providing data to a customer or his or her repository, the utility makes its signature on the data so that any party that knows and trusts the utility's public key can be convinced that the data have been provided by the utility and have not been tampered with. We assume that the utility (and its public key) is trusted by customers as well as third parties. The utility company can be issued a digital certificate from a trusted Certification Authority (CA), and it can be posted publicly on the utility's website.

Third Party can be any party that provides services based on energy usage data provided by customers. For example, service providers that offer energy data analytics, recommendations, or demand-response aggregation may be included. We do not make any trust assumptions on third parties, and therefore data disclosure should be minimal to minimize privacy concerns. Whenever necessary, third parties can verify utility's digital signature for authenticity, which allows them to protect themselves against malicious or fraudulent service requests.

In our framework, utility companies share data only with customers, and data-sharing with third parties always goes through the customer's repository. In other words, customers are required to trust utility companies only in collection of data, not in sharing of data. Another advantage of the lack of direct interaction between the utility and third party service providers is that the utility cannot trace which service providers a certain customer interacts with, which enhances the customer's privacy. The implementation change required on the utility's system is relatively small, and only a signing task needs to be added before it shares data with customers. We describe details of the signing procedure in Section 3.2, and the overhead for it will be shown in Section 4. A utility company still has full access to the customer's data that it collects, and can use it for prediction, anomaly detection [30], and so on, if the customers agree to it in advance. Thus, our scheme does not affect the quality of its services. Likewise, customers can also do data analytics on their own data, if such functionality is supported by a repository. Note that keeping data from utility companies, while it is an interesting future research topic, is not part of our goals.

3.2 Design for Verifiable Redaction of Energy Usage Data

In this section, we present our design for flexible control disclosure of customers' energy usage data while maintaining data verifiability. We employ the idea of Merkle Hash Tree (Figure 2) and tailor it to meet security requirements specific to energy usage data.

3.2.1 Merkle Hash Tree

A Merkle Hash Tree [31] [32] is a tree in which each node stores a hash of some data. Leaf nodes store the hash of the data blocks, and the non-leaf nodes store the hash of the content of their children. Figure 2 illustrates a binary hash tree, but a Merkle Hash Tree does not need to be a binary tree. $H(D)$ is the hash of the data block, and $H(L, R)$ is the hash of the concatenation of the left and right child nodes' content.

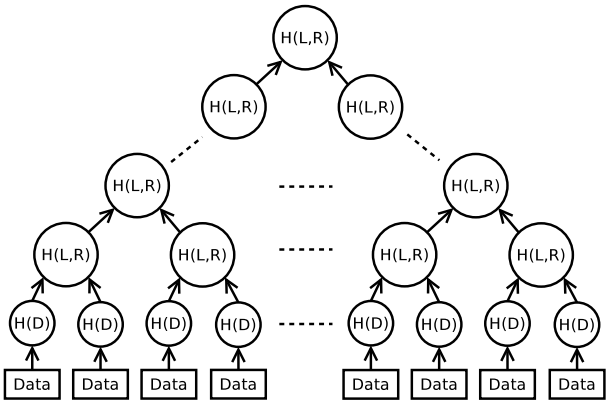

Figure 2: Merkle Hash Tree

Our design of customized Merkle Hash Tree is illustrated in Figure 3. The data block in our design consists of energy usage data along with their metadata, such as units and timestamp. The implication is that the design provides authentication and integrity verification for the metadata as well as the data. The rest of a tree is constructed much like a regular Merkle Hash Tree but with some modifications in hash value calculation, which are explained later in Section 3.2.2. In our scheme, a utility needs to sign the root hash of the tree with its private key to provide authenticity and integrity protection for the data.

Under this construction, any party that trusts the utility's public key can be convinced of the authenticity and integrity of the data. At the same time, each customer can flexibly redact (or hide) an arbitrary portion of the record, without losing the verifiability of the utility's signature, as illustrated

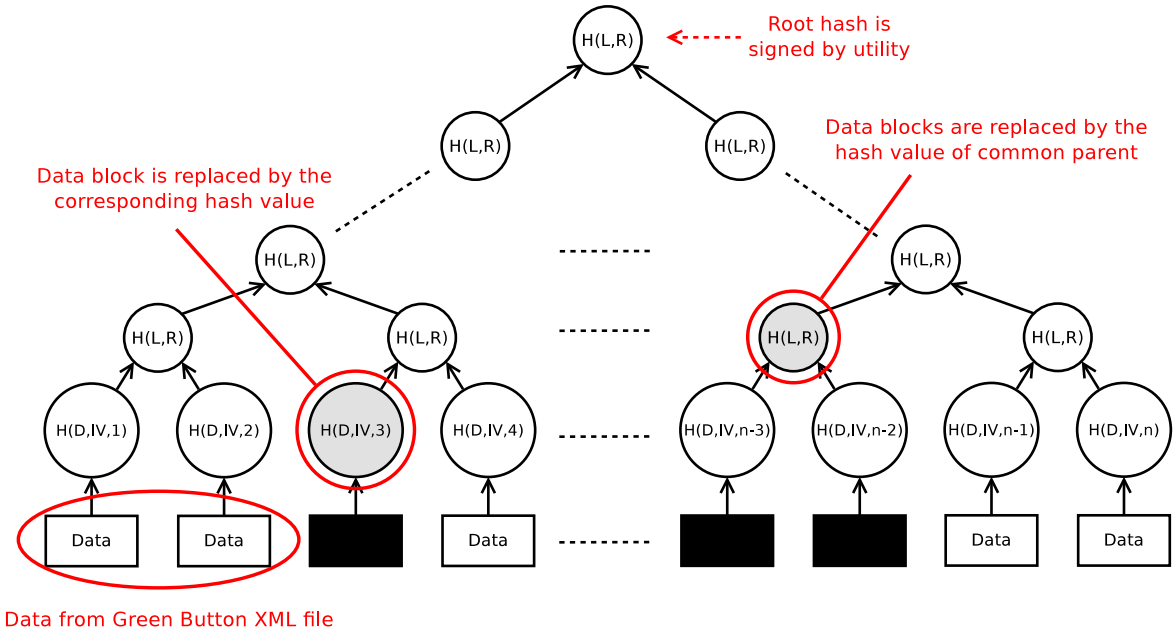

Figure 3: Verifiable, Redactable Energy Usage Data Based on Merkle Hash Tree

in Figure 3. To hide any section of data, we replace it with the hash value of the common parent. It allows the verifier to reconstruct the same root hash value and verify the authenticity of the meter readings that are not redacted. In that way, a customer can selectively disclose the minimal amount of information that he or she thinks is definitely required for the desired service, while still benefiting from third-party services.

3.2.2 *Customization for Energy Usage Data*

While we rely on the notion of redactable digital signatures using Merkle Hash Tree [25], some characteristics of energy usage data and the use case prevent us from simply using regular cryptographic hash functions, such as SHA-256, to calculate the hash tree, as we discuss in this section. Figure 4 shows an example of energy usage data extracted from Green Button data provided by EnerNOC [4]. Some details of Green Button [6] will be discussed in Section 4; for now, we focus on the meter reading that appears under *value* elements.

As can be seen in Figure 4, meter reading values are represented as numbers (in this specific example, as integers), and the values are relatively close to each other. In this example, the maximum difference among the values is just 780. In addition, a third-party service provider would potentially have millions of customers, which likely include many customers with similar energy consumption patterns.

If the hash value is hiding one data block, a third party can use rainbow tables to find the plain text value from the hash values. Considering that search space may be small, it can be very easy to generate such a table and look up the pre-image quickly. Note that just one table would be sufficient for all customers. A similar concern could be raised if multiple values are redacted. Given the collision resistance of secure cryptographic hash functions, the same hash values appearing at the same level of a tree are most likely inserted to mask the same sequences of values below them. In addition, there is a chance of determining a value or sequence of values without the use of rainbow tables if it appears twice or more in the file and one of the values or one of the sequences is hidden and another is present in plain text. An adversary can keep a record of hashes generated by the plain text and match it with the hashes present in the file to determine the actual value. Since the input range is bounded and short, the adversary can create a small rainbow table of its own. It can estimate the values, pre-compute the output hashes, and store them in a database. The hash values will be useful for all the data the adversary receives from all the users. The larger the data sample, the better it is for the adversary as it can match more redacted hash values with the plain text values.

```
<IntervalReading>
  <timePeriod>
    <duration>300</duration>
    <start>1325389500</start>
    <!--2011-12-31 23:45:00 -0400-->
  </timePeriod>
  <value>18706</value>
</IntervalReading>
<IntervalReading>
  <timePeriod>
    <duration>300</duration>
    <start>1325389800</start>
    <!--2011-12-31 23:50:00 -0400-->
  </timePeriod>
  <value>18316</value>
</IntervalReading>
<IntervalReading>
  <timePeriod>
    <duration>300</duration>
    <start>1325390100</start>
    <!--2011-12-31 23:55:00 -0400-->
  </timePeriod>
  <value>19096</value>
</IntervalReading>
```

Figure 4: Sample Energy Usage Data

Taking these observations into account, we made the following modifications to the hash computation. The aim of these modifications is to make it significantly more difficult for third parties to recover redacted values.

1. *Keyed Hash Using a Per-Customer Key*

   To prevent malicious third parties from correlating hash values among multiple customers, we utilize a keyed hash function (e.g., HMAC) with a per-customer key for hash tree calculation. The key is uniquely chosen for each customer and is shared with the utility company and relevant third parties. The utility uses it to construct the Merkle Hash Tree and calculate the root hash. The third parties use it to reconstruct the tree and get the root hash value, which is verified using the utility's public key. For example, we could derive such a shared key by using a customer-chosen PIN code that is utilized in PG&E's system such that only a third party that knows the PIN can access the corresponding customer's energy usage data [12]. However, we should note that in our scheme, such a key is not used for data confidentiality, authentication, or access control.

2. *Random Initialization Vector (IV) and Counters*

   Even with the first modification, the same inputs in the same customer's data provide the same output hash values. An adversary can still take advantage of that to facilitate its effort to break privacy; if it sees the same hash value at the same level of the tree multiple times, the adversary does not have to repeat brute-force calculations. To mitigate that risk, we use a cryptographically secure random number as an initialization vector (IV) to derive a new key for each data block to be hashed. One IV is generated for each data file signed by the utility. Then, within the file, IV is incremented by 1 for each data block, and then XOR'ed with the per-customer key to derive a new key to ensure that no two values in the same file result in the same hash value. Such a counter also prevents unauthorized reordering of data.

The resulting construction is as follows. The hash value computation for the leaf nodes of the Merkle Hash Tree involves data d, key k, and a keyed hash function $M(d, k)$ and is calculated as

$$H(D, IV, i) = M(D, (K \oplus (IV + i))) \quad (1)$$

where D is the data block to hash, K is the key unique to the customer, IV is the initialization vector unique to a data file and i is the counter value for the data block. Hash values for non-leaf nodes are calculated as

$$H(L, R) = M(L||R, K) \quad (2)$$

where L is the hash of the left child node, R is the hash of the right child node, and K is the customer's key.

3.3 Security Discussion

The keyed hash and IV make it hard for a third party to compromise customers' data privacy using strategies that earlier Merkle Hash Tree based scheme were vulnerable to. Selecting a random IV for each data file and incrementing it for each block in the file require a brute-forcing adversary to loop through the range of estimated data values for each hidden value. That further means that, in order to find two hidden data values that match the disclosed parent hash value of them, an adversary has to iterate through all possible combinations of the two values, thereby slowing down the brute-force attack significantly. Security against brute-force attacks will be elaborated below.

Given the modifications we made, an adversarial third party needs to mount a brute-force attack for each redacted leaf node individually to recover the original data. In other words, if 8 meter readings in a row are redacted, to reveal the series of data it would be necessary to perform brute-force attacks on all possible combinations of the 8 data values. If the search space for each is narrowed down to 100 possibilities by means of some educated guess, in the worst case 100^8 ($\approx 2^{53}$) attempts would be required. Note that since cryptographic hash values do not have order-preserving or homomorphic properties, the only way for an attacker to obtain meaningful information about the redacted values would be to identify all the values exactly.

A small number of contiguous data blocks may be vulnerable to brute-force attack. The exact number at which a system could be considered vulnerable would depend on the processing power, storage capacity, and time available to the adversary. If only 4 consecutive data values need to be broken with a search space of 100 possibilities, the adversary would require only 100^4 attempts ($\approx 2^{26}$) to find the values corresponding to the hash values. In an implementation using binary a Merkle Hash Tree, the number of redacted blocks represented by a node in the tree is always some power of 2. For instance, redaction of contiguous 19 data blocks may be represented by 3 hash values: one for 16 data blocks, one for 2 blocks and one for 1 block. Hash values representing few data blocks (1 or 2) can be brute-forced. Therefore, the system is secure only when a large number of contiguous data blocks are redacted. Use of the scheme for use cases with a very small number of redacted data blocks may allow the third party to recover the hidden values.

As discussed above, in general, the larger the number of contiguous nodes to be hidden (which are aggregated into one or a small number of intermediate hash values in a Merkle Hash Tree), the more difficult it becomes to retrieve the original data, as the hash value substituting for hidden values would be calculated from a larger number of data blocks. We acknowledge that even with our modification, it is not difficult for a malicious third party service provider to reveal a redacted value when only a single or a small number of data points are hidden. However, in the case of services or analytics on energy usage data, such a situation is very rare. Most cases require records of only few hours in a whole day. In such situations, the number of consecutive values redacted will be big enough to make brute-force attack infeasible. A shorter metering interval for the same time duration could also strengthen the security of the system since the number of leaf nodes for hash calculation would grow.

Finally, let us consider a real-world scenario in which data file containing daily records is logged in 5-minute interval (288 meter readings per day in total). When meter readings corresponding to the first half of the day (144 readings) are redacted, an adversary needs to determine 2 hash values via brute-force: one corresponding to 128 data blocks, and

one for 16 data blocks. Even if the search space for each meter reading could be narrowed down to 100 possibilities, the number of attempts required to compromise a major portion of the hidden data would be $100^{128} (\approx 2^{850})$, which is highly infeasible and can discourage attacks.

4. IMPLEMENTATION

This section discusses the proof of concept of the proposed scheme based on the Green Button data model [6]. Green Button is a customer-centric approach to sharing energy usage data, and it is attracting broad attention in the energy industry. Considering the success of Green Button and the fact that its philosophy of customer involvement matches ours, we decided to use the Green Button data model as the foundation of our scheme. We first overview Green Button and its data model and then discuss its limitations with respect to our goals. We then discuss our prototype implementation using Java.

4.1 Green Button

4.1.1 Overview

Green Button [6] is based on the idea that consumers should be able to securely download their own easy-to-understand energy usage information from their utility or electricity supplier. It is an industry-led effort in response to a White House call-to-action to provide customers with easy access to their energy usage data in a consumer-friendly and computer-friendly format via a "Green Button" on electric utilities' websites.

Green Button provides a standard data format that includes information types like fifteen-minute load profile, hourly load profile, daily load profile for the past month or year, monthly summary data, etc. The data is present in a standard XML format which allows companies to develop services to provide value-added insights, recommendations, and controls for consumers using their energy usage information. Green Button allows customers to receive information through two mechanisms: Green Button Download My Data and Green Button View My Data. Download My Data downloads an XML Green Button format file, while clicking View My Data renders the Green Button file in the browser using a standard XSLT (EXtensible Stylesheet Language) file.

Green Button is gaining popularity. At least 5 utilities, including Pacific Gas and Electric (PG&E) [12] and San Diego Gas and Electric (SDG&E) [13], have implemented it on their websites, 30+ have made commitments, and 50+ companies either have developed or are developing applications based on Green Button. Many Web-based and smartphone applications, like Leafully [8], and EnergyAI^{TM}[2], are using Green Button data to provide specialized services to consumers.

4.1.2 Data Model

The Green Button data format is based on the North American Energy Standard Board's (NAESB) Energy Services Provider Interface (ESPI) XML standard [10]. The standard specifies an XML data structure for storing and communicating energy usage information. The Atom Syndication Format [15], also used by websites for Web feeds and updates, is used to package Green Button data into XML. Each class of energy usage data is present in an $atom:entry$ element in a Green Button file; each entry has a 128-bit Universally Unique IDentifier (UUID). The data for the class is present in the $atom:content$ type within the $atom:entry$ element. There are other element types within $atom:entry$ element that are used to store metadata like the relationship of the data to other data in the file ($atom:link$).

The Green Button data model is inherently hierarchical and includes the concept of collections. Figure 5 presents the UML (Unified Modeling Language) diagram of the Green Button information model, which shows the relationships among classes. The figure illustrates proposed classes along with existing classes. Next, we briefly discuss the roles of relevant existing classes to provide a context for our modifications. Roles presented below are also mentioned in the ESPI XSD (XML Schema Document) file [5].

- *UsagePoint* is the root element of the data model. It is a logical point on a smart grid network at which consumption is either physically measured (e.g., metered) or estimated (e.g., unmetered street lights).
- *ElectricPowerUsageSummary*, as the name suggests, includes the summary of electric power usage. It can include billing period, total cost of the current billing period, cost during the previous billing period, data related to last year's consumption, etc.
- *MeterReading* represents a meter at a usage point. It does not contain any data and links *UsagePoint* with energy usage data values from a meter.
- *ReadingType* specifies the characteristics associated with the readings included in a meter reading. It can include the unit of the data values (e.g., kWh), the currency type associated with the cost, the quality of the reading, etc.
- *IntervalBlock* contains a set of readings, arranged in a time sequence, of the same *ReadingType*.
- *IntervalReading* specifies a value measured by the meter or other asset. It can include the cost, value in units (specified in *ReadingType*), and date and duration of the reading.

4.1.3 Limitations

Green Button has been developed only for information exchange, and therefore it defines only the data model. In other words, authenticity verification and privacy preservation are not in its primary scope.

Currently, customers downloads their energy usage data from the utility website and, to use other online services, they have to disclose the downloaded Green Button data file to a service provider. The service provider has no method of verifying whether the energy usage data provided by a customer have not been tampered with or forged. Since service providers cannot be convinced of the authenticity of the data, the range of possible usage of the data could be limited. For instance, a service provider would hesitate to use the data for accounting or billing purposes.

The concept of a digital signature is not part of Green Button's specification. Even though it could be used with a signing scheme, such as XML Signature [18], the regular digital signature scheme would only allow data sharing in

an all-or-nothing manner. In other words, redaction invalidates the signature, and therefore a customer has to disclose the entire energy usage data in a Green Button data file to the service provider, limiting the customer's ability to control the amount of data disclosed and giving rise to privacy issues, as discussed earlier.

To overcome those limitations, we propose a modification of the Green Button schema. Our enhancement to the schema is discussed in the next section.

4.2 Enhanced Green Button Data Model

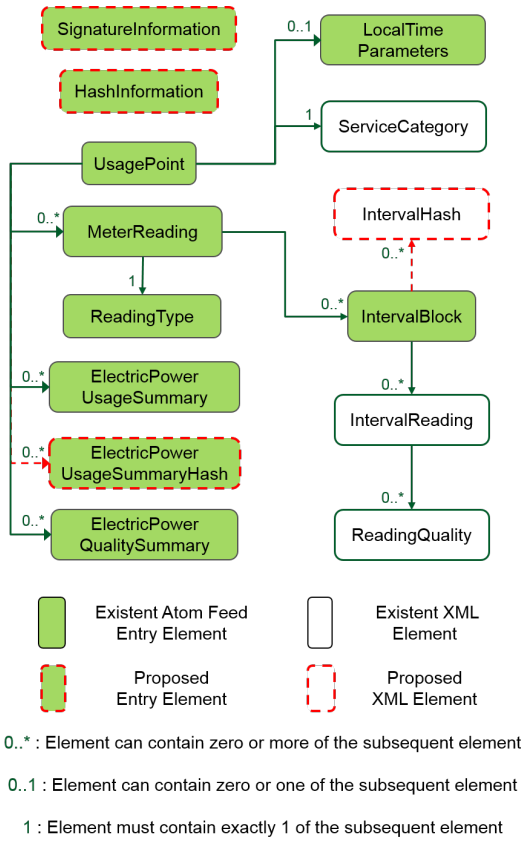

Figure 5: UML Class Diagram of the Proposed Green Button Information Model

Figure 5 presents our additions to the existing Green Button data model, which are explained below. We add 4 data classes to support the design discussed in Section 3.2. A sample XML of each element is presented in Appendix A. Our aim here is to demonstrate one possible, functional implementation to encourage further exploration for such extensions of Green Button.

- *SignatureInformation* contains the signature of the root node of the Merkle Hash Tree and specifies the algorithm used to create the signature (e.g., SHA1withRSA).
- *HashInformation* contains information about the hash algorithm used to construct the Merkle Hash Tree. It includes the algorithm used to create the digest (e.g., HmacSHA1) and the Initialization Vector (IV) value for the file.
- *ElectricPowerUsageSummaryHash* contains the hash value of all power usage summary elements. We believe that the summary of the usage data is sensitive and that the user should have the option of hiding it from the third party if desired. This element is present only if the summary element has been redacted.
- *IntervalHash* contains a hash value used for the Merkle Hash Tree calculation (see Figure 3) when a customer wants to hide a certain portion of the included energy usage data from untrusted parties. It also contains the time and duration of the hidden energy usage data represented by the hash value and the number of blocks hidden by the hash. Such additional information is used to determine the location of the corresponding hash value in the Merkle Hash Tree and is useful in reconstructing the tree for data verification.

4.3 Prototype System

We implemented the entire system in Java 7. To parse and modify the Green Button XML file, we used JAXB 2.2 (Java Architecture for XML Binding). To prevent malicious parties from modifying any sections of the file (e.g., *UsagePoint* values), we included all the data in the Green Button XML file in the Merkle Hash Tree calculation as leaf nodes of the tree (Figure 3). For instance, each value of each element of *ReadingType* is represented as a data block during calculation of the tree. In *IntervalReading*, the data for the *timePeriod*, *cost*, and *value* elements are concatenated (with a delimiter) to form one single data block for the tree. For keyed hash functionality, we used HmacSHA1. It can be easily replaced with any other scheme. We used publicly available standard schema definitions (*atom.xsd*, *xml.xsd*, *XMLSchema.dtd*, and *datatypes.dtd*) along with Green Button's schema definition.

The implementation consists of 3 modules that handle the enhanced Green Button data, including one each for Utility, Repository, and Third Party. Each module's functionality is described below.

- *Data signing module for Utility:* Given Utility's private key and a Green Button data file created in a standardized way, this module calculates Merkle Hash Tree according to the algorithm described in Section 3.2 and makes a digital signature on its root hash value. The module writes the relevant data to the proposed elements *SignatureInformation* and *HashInformation* in the modified Green Button XML file.
- *Data redaction module for Repository:* As inputs, this module takes the Green Button file containing utility's signature and a request from a customer indicating how much information he or she wants to hide. It parses the data file, redacts the data as requested by the customer, and calculates hash values that replace the redacted parts. The module removes values from the *IntervalReading* and *ElectricPowerUsageSummary* elements and inserts *IntervalHash* and *ElectricPowerUsageSummaryHash* elements in the XML file.
- *Data verification module for Third Party:* This module takes the data file provided by Repository, which may be redacted. It calculates the root hash value and verifies it against the utility's signature.

Data (5 min interval)	Original File Size	No. of Interval Readings	Signing Time (ms)	No. of Interval Readings Hidden (%)	Redaction Time (ms)	Redacted File Size (%)	Verification Time (ms)
1 Year	26.5 MB	105,406	5,072	26,351 (25%)	7,566	25.0 MB (94%)	3,174
				52,703 (50%)	9,037	16.7 MB (63%)	2,865
				79,054 (75%)	9,797	8.3 MB (31%)	2,519
1 Month	2.2 MB	8,928	2,447	2,232 (25%)	2,377	2.1 MB (95%)	1,836
				4,464 (50%)	2,339	1.4 MB (63%)	1,717
				6,696 (75%)	2,283	718 KB (32%)	1,500
1 Day	77.6 KB	288	985	72 (25%)	922	77.9 KB (100%)	897
				144 (50%)	921	56.0 KB (72%)	863
				216 (75%)	915	32.4 KB (41%)	798
1 Hour	8.3 KB	12	790	3 (25%)	751	10.5 KB (127%)	756
				6 (50%)	766	9.9 KB (119%)	751
				9 (75%)	746	8.6 KB (104%)	750

Table 1: Prototype Performance

4.4 Performance Evaluation

This section discusses the performance of the three prototype modules. We used a data set provided by EnerNOC [4]. The data consist of 5-minute electricity consumption figures for anonymized commercial and industrial sites for 1 year. For experiments with smaller amounts of data (e.g., 1-month data and 1-day data), we manually extracted part of the 1-year data. HmacSHA1 was used for calculating a Merkle Hash Tree, and SHA1withRSA was used to sign the root hash.

The experiments were conducted on 64-bit Ubuntu 12.04 with kernel 3.2.0-49 and Java 1.7.0_25. The hardware included Intel Core2Duo 2.13 GHz and 4 GB RAM. The performance metrics are presented in Table 1. Each figure in the table is the average of results over 5 different data files of each type. The standard deviation of all the averaged values is less than 5% of the mean.

On a typical desktop PC, the system can complete the signing for 1-year interval data in 5 seconds, redact 75% of the data in 10 seconds, and verify the data in 3 seconds. The time for redaction may seem long, but because it is done on a repository that stores and handles data for a single customer, it is not a performance bottleneck. While the data were being processed, the memory usage was always in the range of 11 MB and 110 MB. Thus, we believe that the system can be practically implemented.

As evident from Table 1, for small files, the redaction time doesn't vary much with the amount of data redacted, although we see some difference for 1-year data. We found that tree calculation and removal of data from the XML consume a major amount of the redaction time.

The hash values present in the redacted file are of nodes at different levels in the tree. As a consequence, data verification takes less time, as the system has to calculate fewer hash values to reconstruct the tree. This effect is more evident in files that have a large amount of redacted data.

For very small files, the redacted files are larger than the original files. The reason is the introduction of the *SignatureInformation*, *HashInformation* and *IntervalHash* elements, which take up more space than the redacted data did.

The IV in our prototype is generated using *SecureRandom* in Java. Sometimes, *SecureRandom* waits to be initialized with a random seed, causing delays in signing the file. In practice, that should not be a problem unless the utility is asked to provide the signed data in real-time. For most of the use scenarios, the Utility can sign the file beforehand and store it.

5. APPLICATION FOR DEMAND-RESPONSE AGGREGATION SERVICE

In this section, to demonstrate one example of a third-party service based on our framework described in Section 3, we discuss the customer-centric design of a DR aggregation service model. However, applicability of the proposed architecture is not limited to it, and another application will be explored in our future work.

At a high level, demand-response aggregation (DR aggregation) service providers, also called DR aggregators, mediate communication between utility companies and electricity consumers who participate in some demand-response program. When a utility wants to curtail electricity demand, it sends a demand-response request (or a demand-response signal) to a DR aggregator. The DR aggregator then sends a curtailment request to a subset of its customers. The customers' performance (i.e., how much they actually succeeded in reducing energy consumption) is reported to and evaluated by the DR aggregator, and, based on the performance, each customer receives an incentive from the DR aggregator. The DR aggregator reports the aggregate amount of curtailment to the utility, which pays incentives according to the DR aggregator's performance and the agreement between the utility and the DR aggregator. Additionally, DR aggregation services often involve incentives for each customer. Therefore, both privacy preservation for customers and data verifiability for DR aggregators are required, and are accomplished under our design.

We next give an overview of commonly used demand-response programs and identify the data required for DR aggregation services. After that, we detail the customer-centric DR aggregation service model based on our proposal.

5.1 Overview of Demand-Response Programs

Nowadays, many utility companies provide either or both of two types of services: CBP (Capacity Bidding Program) and DBP (Demand Bidding Program). Detailed informa-

tion can be found in [35], [34], [39], and [38]. Under CBP and DBP, typically a DR aggregator submits a curtailment commitment to the electricity utility, and sends curtailment requests to DR participants (i.e., end customers) when a demand-response event is announced by the utility. Based on the actual curtailment accomplished and the committed amount, the DR aggregator gets incentives from the utility. One big difference between the two bidding programs is that with DBP a DR aggregator will not be charged a penalty even when it underperforms but with CBP, a penalty may be charged. To avoid a penalty under CBP, a DR aggregator may want to accurately estimate the amount of consumption of each customer based on its past usage data, so that it can minimize the risk of penalty. In both CBP and DBP, to assess the performance of each customer after a demand-response period ends, a DR aggregator needs to know how much each customer curtailed usage during the demand-response event.

There are also other types of demand-response services. Direct Load Control (DLC) allows utility companies (or DR aggregators) to directly turn customer equipments, such as HVAC ON or OFF [36]. DLC is usually agreement-based, making usage prediction and collection of actual energy usage data unnecessary. Dynamic Pricing Model, another popular type of DR program, usually sends just price information to customers and does not need to collect data from them. Thus, in this work, we focus on DR aggregation services under CBP and DBP.

5.2 Data Required for Demand-Response Aggregation Services

In this section, based on the mechanisms of current demand-response programs, we discuss what information is required and sufficient for providing demand-response aggregation services. For both CBP and DBP, one essential requirement is to collect energy usage data to assess each customer's performance. Such performance evaluation is usually done based on the difference between the "baseline" energy consumption and the actual energy consumption of the customer during the demand-response event period. The simple but common strategy for calculating such a baseline is called X-day average baseline [37], which calculates the baseline by averaging the energy consumption patterns of the past X similar days. Given such an assessment procedure, the information that a DR aggregator needs to know is the actual energy consumption during the demand-response event and also the energy consumption data for the same (or similar) time slots on the past X similar days. As mentioned earlier, such energy usage data must be verifiable to a DR aggregator so that it can detect cheating customers or any other malicious behavior.

We also note that, for the performance assessment by DR aggregators, real-time data are not necessarily required. For instance, as long as the DR event period or participant information is appropriately recorded by an aggregator, evaluation and accounting can be done in batch, for example, once a day. Because the frequency of DR events is usually low, (for instance, at most 24 hours a month in the case of CBP by PG&E [34]) that claim holds.

The data for calculating such a baseline can also be used to select customers to whom a DR aggregator sends curtailment requests. Especially in the case of CBP, DR aggregators are motivated to predict which customers will opt in and perform well as accurately as possible to minimize the risk of penalty being charged by a utility and at the same time, minimize the cost for DR aggregators, such as the incentive that needs to be paid to DR participants. Note that to determine such specific set of customers for the DR program, historical data must be collected in advance of a demand-response event.

5.3 Customer-centric Demand-Response Aggregation

Based on the above observations so far, in this section we propose a novel demand-response aggregation service architecture that emphasizes customer centricity. We assume that a utility's smart meters can measure and record electricity consumption with a short enough interval (e.g., 1 minute or 5 minutes) to meet the granularity required for typical DR aggregation services. While measuring with that frequency is not problematic for existing meters, frequent reporting may eat up the network bandwidth of an AMI and a utility's back-haul network, but our scheme does not require real-time reporting. Batch reporting, e.g., hourly or daily, would be acceptable and is practical, as discussed earlier.

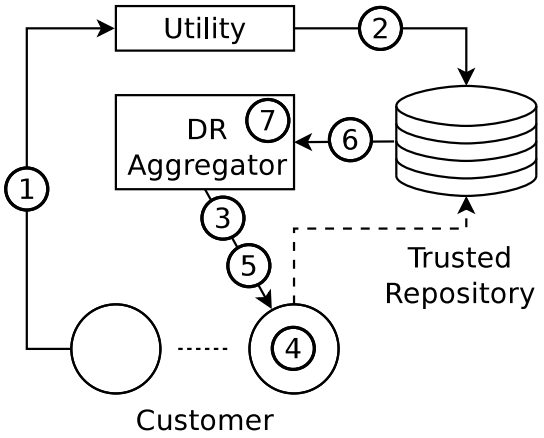

Figure 6: Overview of Customer-centric Model for Demand-Response Aggregation Service

The customer-centric model for demand-response aggregation is illustrated in Figure 6. The architecture is based on the one discussed in Figure 1, and Figure 1's Third Party corresponds to Figure 6 DR Aggregator. We provide the details of the procedure and data flow below.

1. The Utility's meter at the Customer's site sends usage reports periodically (or on a real-time basis, e.g., every 15 minutes) to the Utility. This reporting is done regardless of whether a DR event is active or not.

2. The Customer periodically (e.g., hourly or daily) downloads energy usage data along with the Utility's digital signature on the Merkle root hash value calculated as shown in Figure 3 on demand or at the end of a DR event day. Alternatively, the download can be done automatically by the Trusted Repository. In either case, downloaded records are stored on the Trusted Repository in Figure 6.

3. DR event information (i.e., a curtailment request) is sent to the Customer by the DR Aggregator. For in-

stance, in the case of a "Day-ahead" DR program, such event information is sent the day before the DR event. In the case of a "Day-of" program, information is sent on the day of the event, usually a few hours before the event's start time. The Customer has the option to opt out. Also, if so desired, the DR Aggregator can request historical data from the Customer when performing candidate selection for the DR event. Such information is provided by the Customer as a set of redacted energy usage records. Optionally, to encourage customers to provide more information, the DR Aggregator can offer extra incentives according to the amount of data provided. (For instance, the offer can be "$\$Y$ extra for the last X-day record.") Providing more information to the DR Aggregator may have a higher privacy risk, but such a trade-off can be balanced by the customer.

4. The Customer executes DR (energy usage curtailment), unless he or she opts out.

5. After the DR event ends, the DR Aggregator asks the Customer to send energy usage data. Some historical data for baseline calculation can also be requested by the DR Aggregator in one of the ways mentioned in Step 3. (Requesting historical data at this step might not be necessary if sufficient data were already collected at Step 3.)

6. In response to the usage report request, the Customer sends the minimal amount of energy usage data that corresponds to the DR event period. At the same time, given the incentive offer from the DR Aggregator, the Customer decides how much information he or she feels comfortable providing in exchange for the benefit of the service. Because there is usually no more than one DR event a day, customers' decisions and operations are required no more than a day. Instead of manually selecting data to be disclosed, the customer can define an access control or a data disclosure policy on the repository for automation.

7. After receiving a usage report and data for the baseline calculation, the DR Aggregator first verifies Utility's signature on the energy usage data provided by the Customer. Then, it calculates the baseline, for example by taking the average of energy consumption data, and evaluates the Customer's performance. If Customer redacted too much data and does not meet the minimal requirement as discussed in Section 5.2, the DR Aggregator can suspend the transaction until additional data is provided.

This model does not require installation of an additional metering device by the DR Aggregator, allowing customers to have more control of data disclosure. Also, communication between the Utility and the Trusted Repository and between the Trusted Repository and the DR Aggregator is done using the modified Green Button schema discussed in Section 4, while communication of DR event information distribution (from the Utility to the Customer via the DR Aggregator) can be done by using other protocols, such as OpenADR [11]. To support OpenADR, the Customer may be equipped with an OpenADR client software (i.e., VEN or Virtual End Node) with additional functionality for interacting with the Trusted Repository to automate the process.

6. RELATED WORK

Some research has been done on customer-centric (or consumer-centric) smart grid technologies. In [27, 40], the authors propose a cloud-based approach to deploying data analytics functionality in a secure, privacy-enhanced manner. At a high level, in their architecture, consumers own and manage their own virtual machine in a cloud, called *VHome*, to store energy usage data and run analytics. However, the verifiability of shared data is entirely outside their scope, and therefore their scheme is not suitable for services like demand-response aggregation that require trustworthy data for billing and accounting purposes. In addition, their scheme needs to involve IaaS providers as well as VHome SaaS providers in addition to customers and the utility company. Our scheme has a simpler interaction model and fewer trust assumptions.

The trusted repository in our scheme may seem similar to the energy data center proposed in [26]. However, the main purpose of the energy data center is to consolidate energy usage data of many customers to facilitate sharing of anonymized data among utility companies, customers, and third parties, and therefore is not customer-centric. In addition, security and privacy guarantees, including customer consent and protection against cyber attacks, rely largely on contracts among involved organizations, which are not systematically enforced. Our goal is to provide security and privacy guarantees in a cryptographic manner, and control over such functionality belongs to customers.

Green Button Connect My Data is a service that allows a customer to authorize a third party to access his or her Green Button data directly from a utility's website. The service doesn't allow customers to provide selected data to a third party. The mechanism is based on OAuth 2.0 [7] and allows the third party to automatically collect all of a customer's energy usage data from a utility without any further intervention from the customer. The customer has the power to revoke the access, but that would prevent the customer from accessing the services of the third party. The scheme itself is still in development, and very few utility companies have implemented it, SDG&E being one of them [14].

Use of a Merkle Hash Tree for redactable signatures is done in a number of different domains. For instance, in identity management area, it is used to implement minimal-disclosure identity credentials, which allow a credential owner to flexibly hide part of identity claims without losing the verifiability of the identity provider's digital signature [19]. In a Web service setting, [21] proposes the use of it in UDDI authentication. However, to our knowledge, our work is the first application in energy usage data management. Moreover, we tailored the scheme to meet the security and privacy requirements that are unique to this context.

Customer-centric data-sharing has also been explored in other domains. For instance, in the identity management area, a number of user-centric approaches have been explored [22, 29]. The schemes have been proposed to mitigate the drawbacks of identity provider-centric approaches that not only give owners of identity credentials little control over their digital identity, but also could cause privacy concerns by allowing the central provider to track the owners' activities in cyberspace. The same trends can be seen in the e-healthcare domain. The emergence of Personal Health Record (PHR) systems [9, 33] enhances patients'

control over their own data and privacy in a similar way. This situation is analogous to that of energy usage data. In an approach like Green Button Connect My Data, a utility company, which can be seen as a central entity, can potentially know which third-party service providers a customer is interacting with. Our system can be seen as a solution to minimize such privacy concerns.

7. CONCLUSION

In this paper, we presented a customer-centric framework that enables management and sharing of energy usage data, extending the well-accepted Green Button data model [6] in two ways. In order to enhance customer privacy, our scheme allows redaction of an arbitrary portion of Green Button data when a customer shares his or her data with a third-party service provider. The extended schema is still compatible with the system using the standard schema, such as visualization using Green Button View My Data [6]. On the other hand, the presented data, even when some part is redacted, can be verified and authenticated by any party, which makes it possible to reliably use the Green Button data even for billing and accounting purposes. We also discussed a model of demand-response aggregation service based on the proposed framework.

One of the major components of our future work is integration of the proposed mechanisms into a working system. We are currently in the process of implementing a prototype of the customer-centric demand-response aggregation service. We will also explore other applications, besides demand-response aggregation services, that can be implemented on top of our customer-centric model. Some services, such as sophisticated recommendation services, need to use statistical or machine-learning techniques, which may require different privacy-preservation schemes. Another direction would be a mechanism or user interface that assists customers' decision making to appropriately balance privacy and enable them to benefit from services.

8. ACKNOWLEDGMENTS

We greatly thank Dr. Martin J. Burns for sharing the latest Green Button XML schema for our prototype implementation and evaluation.

9. REFERENCES

[1] ECS. http://www.ecsgrid.com/.
[2] EnergyAI^{TM}. http://www.energyai.com/.
[3] EnerNOC. http://www.enernoc.com/.
[4] EnerNOC Green Button data. http://open.enernoc.com/data/.
[5] ESPI XML Schema Definition file. http://naesb.org/copyright/espi.xsd.
[6] Green Button. http://www.greenbuttondata.org/.
[7] Green Button Connect My Data Authorization Reference. http://osgug.ucaiug.org/sgsystems/OpenADE/Shared%20Documents/Testing%20and%20Certification/GreenButtonTestPlan/referenceMaterial/GreenButtonAuthorization.docx.
[8] Leafully. https://leafully.com/.
[9] Microsoft HealthVault. http://healthvault.com/.
[10] NAESB ESPI Standard. http://www.naesb.org/ESPI_Standards.asp.
[11] OpenADR Alliance. http://www.openadr.org.
[12] Pacific Gas and Electric (PG&E). http://www.pge.com/myhome/myaccount/using/thegreenbutton/.
[13] San Diego Gas and Electric (SDG&E). http://www.sdge.com/green-button.
[14] San Diego Gas and Electric's (SDG&E) Green Button Connect My Data service. http://www.sdge.com/using-green-button-connect-my-data.
[15] The Atom Syndication Format. http://tools.ietf.org/html/rfc4287.
[16] The Laws of Identity. http://www.identityblog.com/stories/2004/12/09/thelaws.html, 2004.
[17] Opower Lands Utilities PG&E, BG&E. http://gigaom.com/2011/04/07/opower-lands-utilities-pge-bge/, 2011.
[18] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon. XML Signature Syntax and Processing Version 1.1. http://www.w3.org/TR/2013/REC-xmldsig-core1-20130411/, 2013.
[19] D. Bauer, D. M. Blough, and D. Cash. Minimal information disclosure with efficiently verifiable credentials. In *Proc. of ACM CCS2008 DIM Workshop*, pages 15–24, 2008.
[20] D. Bergman, D. Jin, J. Juen, N. Tanaka, C. Gunter, and A. Wright. Nonintrusive load-shed verification. *Pervasive Computing, IEEE*, 10(1):49 –57, Jan.-Mar. 2011.
[21] E. Bertino, B. Carminati, and E. Ferrari. Merkle tree authentication in UDDI registries. *International Journal of Web Service Research*, 1(2):37–57, 2004.
[22] David Chappell. Introducing Windows CardSpace. http://msdn.microsoft.com/en-us/library/aa480189.aspx.
[23] G. Hart. Nonintrusive appliance load monitoring. In *Proc. of the IEEE*, volume 80, pages 1871 –1891, Dec. 1992.
[24] T. Iwao, K. Yamada, M. Yura, Y. Nakaya, A. A. Cárdenas, S. Lee, and R. Masuoka. Dynamic data forwarding in wireless mesh networks. In *Proc. of The First IEEE International Conference on Smart Grid Communications*, pages 385–390. IEEE, 2010.
[25] R. Johnson, D. Molnar, D. X. Song, and D. Wagner. Homomorphic signature schemes. In *Proc. of CT-RSA*, pages 244–262, 2002.
[26] A. Lee and M. Zafar. Energy data center. http://www.cpuc.ca.gov/NR/rdonlyres/8B005D2C-9698-4F16-BB2B-D07E707DA676/0/EnergyDataCenterFinal.pdf, 2012.
[27] W.-H. Liu, K. Liu, and D. Pearson. Consumer-centric smart grid. In *Proc. of The IEEE PES Conference on Innovative Smart Grid Technologies (ISGT) 2011*, pages 1–6, 2011.
[28] D. Mashima and M. Ahamad. Enhancing accountability of electronic health record usage via patient-centric monitoring. In *Proc. of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 409–418, 2012.
[29] D. Mashima, D. Bauer, M. Ahamad, and D. Blough. User-centric Identity Management Architecture Using

Credential-holding Identity Agents. *Digital Identity and Access Management: Technologies and Frameworks, IGI Global*, pages 78–96, 2011.

[30] D. Mashima and A. A. Cárdenas. Evaluating electricity theft detectors in smart grid networks. In *Proc. of The 15th International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 210–229. Springer, 2012.

[31] R. C. Merkle. Protocols for public key cryptosystems. In *Proc. of IEEE Symposium on Security and Privacy*, pages 122–134, 1980.

[32] R. C. Merkle. A certified digital signature. In *Proc. of CRYPTO*, pages 218–238, 1989.

[33] A. Mohan, D. Bauer, D. M. Blough, M. Ahamad, B. Bamba, R. Krishnan, L. Liu, D. Mashima, and B. Palanisamy. A patient-centric, attribute-based, source-verifiable framework for health record sharing. *CERCS Technical Report, Georgia Tech*, (GIT-CERCS-09-11), 2009.

[34] PG&E. Capacity bidding program (CBP). `http://www.pge.com/mybusiness/energysavingsrebates/demandresponse/cbp/`.

[35] PG&E. Demand bidding program (DBP). `http://www.pge.com/mybusiness/energysavingsrebates/demandresponse/dbp/`.

[36] PG&E. SmartAC program. `http://www.pge.com/en/myhome/saveenergymoney/energysavingprograms/smartac/index.page`.

[37] SCE. 10-day average baseline and "day-of" adjustment. `http://asset.sce.com/Documents/Business%20-%20Energy%20Management%20Solutions/10DayAvgBaselineFS0811.pdf`.

[38] SCE. Capacity bidding program. `http://www.sce.com/cbp/capacity-bidding-program.htm`.

[39] SCE. Demand bidding program. `http://www.sce.com/dbp/demand-bidding-program.htm`.

[40] R. P. Singh, S. Keshav, and T. Brecht. A cloud-based consumer-centric architecture for energy data analytics. In *Proc. of ACM E-Energy 2013*, pages 63–74, 2013.

APPENDIX

A. PROPOSED XML ELEMENTS

Here we present a sample of the XML elements we added to the standard Green Button schema. The purpose of these elements is discussed in Section 4.2. The *id* type in the elements is a UUID for the element.

```
<entry>
    <id>urn:uuid:55673B87-3009-4B05-B5BD-C9E841C7B154
    </id>
    <content type="">
        <ns2:SignatureInformation>
            <ns2:SignatureValue>2a444b90bdd8fdeb07352
            49cf7169826588c8a85e5ea5d090e7e332b91f580
            a71620f50d4a05ee88facec0857cfd125c01209fa
            c642006ebb8f76b82c4e5bc42
            </ns2:SignatureValue>
            <ns2:SignatureAlgorithm>SHA1withRSA
            </ns2:SignatureAlgorithm>
        </ns2:SignatureInformation>
    </content>
</entry>
```

Figure 7: *SignatureInformation* Element

```
<entry>
    <id>urn:uuid:CF2BB894-FAAF-4408-95C2-559091C0D0D0
    </id>
    <content type="">
        <ns2:HashInformation>
            <ns2:HashAlgorithm>HmacSHA1
            </ns2:HashAlgorithm>
            <ns2:InitializationVectorValue>8ea31f5b3d
            d56ec30ebd014c2da10029aeb34830974d69d5dae
            d34ce85a179fdf936c043d5c581823542671913d4
            b4714fe2a96a756b9a035998b3d3127987eb
            </ns2:InitializationVectorValue>
        </ns2:HashInformation>
    </content>
</entry>
```

Figure 8: *HashInformation* Element

```
<ns2:IntervalHash>
    <ns2:timePeriod>
        <ns2:duration>9830400</ns2:duration>
        <ns2:start>1335192900</ns2:start>
    </ns2:timePeriod>
    <ns2:value>b00ab83b5c2cbc222f472ede0bd73520d1bdbb
    0c</ns2:value>
    <ns2:hiddenBlocks>32768</ns2:hiddenBlocks>
</ns2:IntervalHash>
```

Figure 9: *IntervalHash* Element

```
<entry>
    <id>urn:uuid:5746036f-3d85-4cdb-a2be-c70a0e53a30f
    </id>
    <link rel="self" href="RetailCustomer/41/UsagePoi
    nt/01/ElectricPowerUsageSummary/01"/>
    <link rel="up" href="RetailCustomer/41/UsagePoint
    /01/ElectricPowerUsageSummary"/>
    <title>Usage Summary</title>
    <content>
        <ns2:ElectricPowerUsageSummaryHash>
            <ns2:value>7338865b07eccd5524bb5cd9230ae9
            bf50f65f94</ns2:value>
        </ns2:ElectricPowerUsageSummaryHash>
    </content>
    <published>2013-04-04T03:35:38Z</published>
    <updated>2013-04-04T03:35:38Z</updated>
</entry>
```

Figure 10: *ElectricPowerUsageSummaryHash* Element

- *SignatureInformation* contains two elements: *SignatureValue* and *SignatureAlgorithm* (Figure 7). *SignatureValue* stores the signature value of the signed Merkle root hash value. *SignatureAlgorithm* specifies the cryptographic algorithm identifier that the data issuer uses when signing.
- *HashInformation* contains two elements: *HashAlgorithm* and *InitializationVectorValue* (Figure 8). *HashAlgorithm* specifies the keyed hash algorithm used to calculate the Merkle Hash Tree. *InitializationVectorValue* contains the IV for the file as explained in Section 3.2.2.
- *IntervalHash* stores the hash value of a node in the Merkle Hash Tree (Figure 9) that corresponds to redacted data. *timePeriod* stores the time period of a single or a sequence of redacted child leaf nodes. The hash value is stored in *value*, and *hiddenBlocks* contains the number of data blocks redacted by this element.
- *ElectricPowerUsageSummaryHash* stores the hash value of XML data under the *ElectricPowerUsageSummary* element (Figure 10) when the contents of *ElectricPowerUsageSummary* in the original XML need to be hidden. The *id*, *link*, and *title* values are taken as is from the summary element.