Migrating Building Energy Management System to Public Cloud in a Secure Way

Eun-Kyu Lee and Iksoo Kim

Dept. of Information and Telecommunication Engineering,
Incheon National University, Incheon, Korea
{eklee, iskim}@inu.ac.kr

Abstract. An energy management system in a commercial/industrial building (BEMS) plays an important role as a power consumer in a smart grid. But, many building facilities cannot afford the system, and a cloud service emerges as an alternative. As known, cloud services still struggle with security problems, which prevents the BEMS from being widely deployed on the cloud. This paper addresses security concerns of confidentiality, integrity, and availability in the cloud BEMS by using attribute based encryption and a cloud computing service publicly available. First, the encryption scheme assure not only scalable but also confidentiality even though the cloud is not reliable because every usage data is encrypted. The cloud Service provides security, thereby hiding the server location. Last, the cloud ensures availability. Whenever the cloud supports connections from user to server, and our implementation allows reading data from every device. Thus, when we used together, the encryption and the cloud provide the best implementation for secure smart grid computing.

Keywords: smart grid, energy management system, building, cloud, security.

1 Introduction

A smart grid is referred to as modernization of an existing power system by integrating the state-of-the-art information technology. Its reliable operation heavily depends on a great amount of data produced by various sensor systems in the customer domain. Storing and processing the data in a centralized manner, therefore, show limitation. A cloud computing can be considered as an alternative distributed system. In this work, Consumer Electric Usage Data (CEUD) is stored and managed on the cloud and accessed by users. Decentralization eliminates a potential bottleneck of performance. Furthermore, the cloud promises the benefit of data availability. Due to advanced data management schemes applied to the cloud technology, data is accessed ubiquitously and is less stressful with denial of service attacks. However, these benefits do not come without challenges. As private data resides on a public area, data protection must be the highest priority. However, the decentralization property makes it hard to apply conventional data protection algorithms directly.

Iksoo Kim is the corresponding author.
The main challenges that any securing system seeks to address are confidentiality, integrity, and availability. Below, we describe each of these in detail and explain how they relate to smart grid implementation. Confidentiality denotes that all of the data can be exposed to the threat of hacking even though the data is stored in the cloud. People want to store data securely. This is called confidentiality. Especially, in the smart grid, usage data should be assured confidentiality because the data can be sensitive information in privacy. Therefore, the smart grid should be confidential. Integrity is that data should not be altered by falsifying, damaging, and errors. Also, data can be changed by a hacker or user’s mistake. All of the smart grid data should be assured integrity like logs. The data is used for analyzing and compiling statistics. If the data is changed, the whole of the smart grid system cannot work exactly. Thus, the smart grid system should assure integrity. Availability is the degree that includes reliability and maintainability. This can be shown the probability that the system and device maintain their function at particular time. In the smart grid, availability is important point because electricity is always provided from electric utility companies. People should be available to read and control their electricity status whenever they want to access.

To solve the problems, this paper takes Ciphertext-Policy, Attribute-Based Encryption (CP-ABE) that is a type of identity-based encryption [1]. There is only one public key, and everyone can share the key. Also, there is a master private key used to make more restricted private keys. Moreover, CP-ABE is attribute-based encryption. Users’ private keys are given list of “attributes,” and files are encrypted under “policy” over those attributes. These files can only decrypt if attributes satisfy policy. Therefore, CP-ABE has two major advantages for our project. First advantage is that encrypted files for untrusted storages, and the other advantage is that setting up keys is offline. Hence, CP-ABE assures confidentiality and integrity.

Fig. 1. The proposed system consists of three components: an ESI, a Cloud BEMS, and a client.

2 Proposed System

The proposed system consists of 3 participating components as shown in Fig. 1: an ESI (Energy Service Interface), a Cloud BEMS (Building Energy Management System), and a client. An ESI consists of energy-measuring devices (e.g., power strips and smart meters) and a gateway system. The gateway communicates with devices using ZigBee communications technology and collects energy usage of the devices.
periodically. Also, the gateway provides data service (the energy usage) to smart grid network via XML web service. A Cloud BEMS consists of two modules, storage and security. Storage receives energy usage data from Gateway, and security performs access control using ID and password. These are on Cloud in order to assure availability. A client represents any system that consumes energy data measured at the building. It can be an electric utility company or a 3rd party service provider. The client does not connect to gateway directly. Instead, it can connect to cloud and read usage data.

The implementation design of the proposed system is shown in Fig. 2. The ESI gateway and the Cloud BEMS is connected by oBix server and the client. Also, Crypto module is CP-ABE that encrypts usage data. Thereafter, the encrypted data is stored to MongoDB that is no SQL database. The client sends a request to the Cloud BEMS in order to get usage data. At that time, Google Protobuf enacts protocol between the Cloud BEMS and the client. They exchange encrypted data by JSON. After that, the client decrypts the data by CP-ABE as the user’s attribute. Finally, the client shows the data by its own Java GUI.

The Cloud BEMS implements the followings so as to function a secured energy management. (1) MongoDB is a scalable, high-performance, open source, document-oriented database and typical no SQL database. In our project, we do not use RDBMS because this server-client does not require complex query and relational database. Therefore, MongoDB has better performance than RDBMS. (2) Jetty provides an Web server and javax.servlet container, plus support for Web Sockets, OSGi, JMX, JNDI, JASPI, AJP and many other integrations. We decide Jetty server for our project. (3) JSON (JavaScript Object Notation) is a Lightweight data-interchange format. This is simple and easy to use for data communication. We use Json for data communication. (4) Proto buffer, Protobuf from Google, is a way of encoding structured data in an efficient yet extensible format. Google uses Protocol Buffers for almost all of its internal RPC protocols and file formats. This is implemented in many popular programming languages; one of them is Java. Through use of Java Reflection we can encode the same protocol using XML, JSON, Smile and so on. Also, adding new fields does not break compatibility.

The client implements a CP-ABE decrypt module, oBix handler, and Java GUI. It reads usage data from server, and then decrypts the data. Java GUI shows us the amount of data usage as the authentication of the user by graph.

Fig. 2. Components of the proposed system are implemented so as to communicate.
3 System Operations and Performance Evaluation

The proposed system operates in the following steps. First, the BEMS on the cloud pulls data from the ESI gateway periodically. This value is configurable and is currently set to a data pull every 10 minutes. This data read is performed by means of oBIX, wherein the client in the cloud sends requests to the server in the gateway in order to read usage data. After this process, the usage data is encrypted by CP-ABE. This relates the encrypted data to a corresponding authentication policy. Next, any user of the utility can sign in to the cloud server to read the encrypted data. If the user is authorized to access the data, the user can decrypt the data by using their private key. This will then let him view the usage data that he wants to access.

![Fig. 3. Iterative experimentation of both encryption and decryption processes.](image)

Experimentation and Evaluation. We note that an entire testbed has been built in a real-world laboratory environment. We run experiments on top of the testbed in order to evaluate the effectiveness of the proposed system. In particular, we focus on the encryption and decryption overhead that our testbed uses the ABE. We ran our encryption module on a typical XML file, which we expect our system to encounter, 10 times and the findings are reported below. For this experiment, we used a 64-bit Core 2 Duo, 1.2 GHz Intel Centrino running a 32-bit Linux-based system. Fig. 3 illustrates the results, processing time of encryption and decryption. As shown, encryption is quite slow. We encrypt a file of size 563 bytes, and it takes around 3 seconds to encrypt. This delay may turn out to be unacceptable in real systems. With respect to the decryption, we use a private key with the following attributes: BLDG_ENG, S_TIME, E_TIME and READ, with them being the BEMS Name, the start and end time stamp of the request and the authorization for the user. We ran around 10 tests of this too and found the decryption time to be around 350ms, which is also shown in Fig. 3.

References