A Smart Energy System with Distributed Access Control

Cheng-Ting Lee*, Cheng-Hsun Yang*, Chun-Min Chang*, Chung-Yi Kao*, Hua-Min Tseng*, Henpai Hsu*, Pai H. Chou†*
*Department of Computer Science, National Tsing Hua University, Taiwan
†Department of Computer Science, University of California, Irvine, USA
sunkist0704@hotmail.com, {assassin784533, chun.m.chang, backman.only, thm822, dreamfliper, pai.chou}@gmail.com

***Abstract*—This paper presents a new smart energy (SE) system with distributed access control. Many other SE systems to date support remote control and automation but not access control, making them applicable to homes but not offices or other public settings. Even if they implement access control, virtually all existing SE systems suffer from central point of failure. To address these problems, we propose a new SE system that supports not only access control but also distributed access from multiple devices without relying on centralized control. In addition to access-control list, we also take advantage of the proximity tag feature supported by the Bluetooth Low Energy (BLE) protocol without requiring users to take out their identifying BLE devices from their pockets as one would have to do with RFID tags. Experimental results show that our SE system to offer the security features required for public deployment with minimal power, latency, and cost overhead.**

I. Introduction

Smart energy (SE) systems have been one of the most widely studied classes of applications for the Internet of Things (IoT). They predated the Internet and are still in active development today. They control everything from lighting and appliances to HVAC (heating, ventilation, and air conditioning) systems. SE has the potential of significantly reducing energy waste without sacrificing the level of convenience and comfort.

However, despite decades of evolution, SE still has been limited to hobbyists, fancy hotels, and some high-end apartments, while the great majority of the homes have yet felt the impact of such kind of technology. Moreover, most public environments have yet to adopt SE, other than occupancy sensors in offices and light sensors for outdoor lights. We believe that besides the mismatched user interface, a very important reason for the lack of wide adaptation is that most SE systems suffering from the lack of access control and central point of failure.

A. Access Control

Access control means the ability to permit or deny accesses to the SE system according to some policy. Access control can be based on the identity of the user or the owner of a device, the possession of an authorization token, or a combination of various conditions. Access control is not necessary within a home environment, because members of the same phone can access all the conventional physical switches, and there are no compelling reasons for access control in the SE version. Many SE systems allow access from a remote site over the Internet, in which case access control in the form of a shared key is usually used. Similarly, given the recent popularity of smartmobiles (i.e., smartphones and tablets) as a user-friendly software remote control, access control usually simply leverages the Wi-Fi network password as its only form of access control.

Access control is mandatory for SE systems that are deployed in an office or a public environment. This is because such an environment requires administration to function properly. For example, in a shopping mall, shoppers should not be allowed to turn off lights or ventilation systems at will. Although it is possible to put these SE system on its own separate (wired or wireless) network from the public, network-based access control is still not enough. For example, in an office environment, workers in adjacent offices are likely to be on the same network (e.g., Wi-Fi), but they should not be allowed to access those power switches in another colleague's office unless explicitly permitted. At the same time, a facilities manager of the building should be able to access all SE devices. Moreover, the manager should be able to grant and revoke access rights to individual occupants of the office, and users may be able to further grant access to their assistants. Unfortunately, these features are not supported by most of today's SE systems.

B. Central Point of Failure

Another problem with many of today's SE systems is that they rely on centralized control, which has the problem of central point of failure. That is, if the central device fails, the entire system can be rendered useless. Centralized control is a natural organization for SE systems that require network bridging. For example, when the user is on a smartphone that attempts to control a ZigBee-based SE system, most likely the command will need to be sent over Wi-Fi to a gateway device that translates the command to ZigBee. A centralized structure is convenient for implementation, because the user can get a global view of all device states within the SE network. Even if the central point of failure does not lead to disastrous consequences (e.g., the lights and devices may still be controllable via physical access), centralized access control can fail, as an authorized user who presents a valid key may fail to authenticate due to central failure. However, if the policy is to grant access during central failure, an unauthorized user may be able to gain access during the brief downtime of the system, making it easy to attack.

C. Distributed Access Control over BLE

To support access control without central point of failure, we propose a distributed access control scheme. First, it requires a transport that can connect the command-issuing device (e.g., sensor, smartmobile) to the target device (e.g., light switches, actuators) without a central gateway device. Second, the access control scheme must be able to work by the individual devices involved in the transaction. Note that our scheme can still make use of a gateway or some central device just like other SE systems; it just means that our system will continue to operate even if such a device fails.

Our approach to access control is that we distribute the role-based access-control list to individual devices so that they have the necessary information to control access individually. We further consider two modes of distributed access control: one for electronic control and the other for physical control. Electronic control means that a device such as a smartmobile, a PC, or a gateway device issues or relays a command to the target node. Access control is accomplished by requiring a credential along with the command. Physical control means the user physically contacts a switch or a dial to change the device's state. In this case, access control takes advantage of the proximity tag feature commonly implemented using BLE. That is, the user is required to carry one or more registered devices that must be in sufficient proximity in order to be granted access to the particular command. This is similar to swiping an RFID tag before being allowed access to a locked door or an elevator. The advantages with BLE implementation include (1) much cheaper cost than RFID readers (2) no need to take out the tags from the pocket (3) any BLE device can serve as tags without requiring dedicated tags (4) conjunctive and disjunctive tags can be configured as keys.

II. Background and Related Work

SE solutions have been proposed for a variety of wired and wireless protocols. This section provides a brief survey of related systems.

A. SE Systems

The most well known contenders in this space include X-10, ZigBee, Z-Wave, Insteon, and DALI. X-10 is the oldest that uses power-line communication (PLC). It is simple to set up, as it requires no additional wiring, but PLC is not always reliable, offers no security, and no access control. ZigBee [1]–[3] is a general-purpose protocol originally invented for wireless sensor networks (WSN) but also implements profiles for home lighting control and smart energy. It is used in smart meters that can communicate with appliances such as refrigerators and washers to optimize for time of use. ZigBee is also used in some lightbulbs. Z-Wave [4], unlike ZigBee, is invented primarily for smart home, lighting, appliances, curtain control, and many more. Insteon [5] combines an X10-compatible PLC protocol and an RF mesh network protocol. DALI, for Digital Addressable Lighting Interface, is a wired standard for controlling lighting, motor control for curtains, motion detectors, and emergency monitors.

B. Access Control

One thing all of these protocols have in common is that, despite mesh networking capabilities, they fall short of access control between peers, and their access control schemes are limited to the central controller [6]. None of them are directly compatible with smartmobiles, and all require either a gateway [7] or a dongle. Because a dongle is inconvenient and incurs extra power, a gateway becomes the preferred option and is commonly assumed by designers of SE systems to be a necessary component. When the application demands access control, the gateway becomes the natural place to implement it, because it abstracts away the physical connectivity from the access control features: the issuer of a command can send it over any protocol that the gateway can handle, whether via web service over the Internet or locally via any bridged wired or wireless protocols.

Unfortunately, making the gateway such an important component can result in the problem of central point of failure. That is, a malfunctioning gateway can render the entire SE system inoperable. Our system can avoid this kind of problem because we retain the distributed characteristics. Even if some nodes are down, the other nodes can continue to work.

C. Bluetooth Low Energy (BLE)

Bluetooth 4.0 Low Energy (BLE) Technology is a subset of the Bluetooth 4.0 protocol. It can work as either master-slave or broadcast. Although the protocol does not support mesh topology or multi-hop, it can be implemented in the application layer if necessary. One main reason we consider BLE is its direct compatibility with smartmobiles and many emerging IoT devices. This is an important property as it enables not only the infrastructure devices to talk to each other but also directly compatibility with smartmobiles without relying on a gateway or a translator. Although other protocols such as ZigBee could also use a dongle for protocol bridging without a centralized control, doing so instead of using a built-in interface is extremely cumbersome and is unlikely to become popular, since dongles can break easily, not to mention the added cost. On the other hand, protocol features such as multi-hop and other profiles can be added easily as part of the app and in the firmware without any material cost, making them much easier to adopt by the users.

More importantly, direct interactivity enable existing BLE devices to double as RF-based *proximity sensors* with each other at the cost of a few KB of firmware upgrades without any hardware cost. Proximity features cannot be implemented by protocol bridging. Although Wi-Fi Direct can also be another candidate protocol for IoT as a direct contender with similar direct compatibility, Wi-Fi consumes significantly more power and the modules are much more expensive. The lowest-cost single-chip MCU with integrated BLE RF is US$1.06 in 2K quantities at the time of this writing, compared to about US$3 for a Wi-Fi chip only without the MCU. Therefore, the applications of Wi-Fi Direct may be more specialized.

In terms of energy efficiency, Wi-Fi is well optimized for bulk data transfer, at 5.25 nJ per bit, whereas BLE is about 153 nJ per bit. However, BLE is optimized for transmitting *state* whose update frequency is on the order of once per second or longer. Its peak current is lower than Wi-Fi by

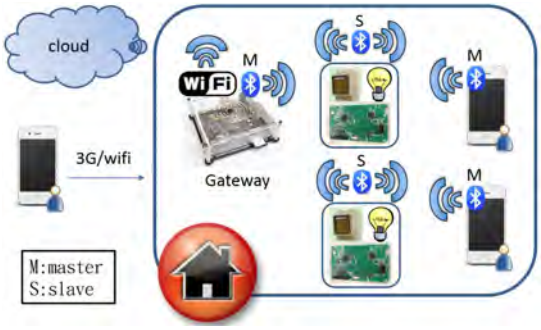

Fig. 1: System overview.

an order of magnitude, and its average current is even lower, thanks to the duty cycling concept called connection interval in BLE.

When a BLE master and a slave pair with each other, they agree on a frequency-hopping sequence and the connection interval. That is, the master and slave agree to send and receive data on a specific channel and switch channels according to a sequence over time. The connection interval is the amount of time the paired devices agree to exchange application-layer or link-layer data to maintain the connection, from 7.5 ms to 4 s. Another concept is *slave latency*, which is the number of connection events that a slave can ignore as a way to further save energy. These parameters enable the system designer to make trade-offs between responsiveness and energy saving.

III. System Overview

Our BLE-based SE system consists of the following subsystems: one or more nodes, one or more terminals, and an optional Internet gateway. Fig. 1 shows an overview of our system.

A. SE Nodes

A *node* in our SE system is one of several types of devices on the wireless (BLE) network, including light switches, power sockets, various sensors, actuators, and tags.

1) Switch vs. Button: We make a distinction between a switch and a button. A switch in SE context is an electronic device for controlling electrical connection. A button, on the other hand, is a user-input device for triggering an SE event, which may be to turn-on, turn-off, or others. In a traditional light switch, a button is integrated with a switch, but in our SE system, the switch can be either integrated or detached from a button. Moreover, a switch may be integrated with a power outlet.

A button can be a stand-alone node without being integrated with a switch. One reason for this is that this allows us to create *macro buttons*, where events triggered by a button is not hardwired to a switch but can be programmed to any event that makes sense. One example is an all-on, all-off macro near the entrance of a home. When pressed, it sends events to a list of other switches that should be turned on or off. It can also be a button in multi-way upstairs-downstairs light switch.

2) Sensors and Actuators: A sensor node contains a sensing device that detects conditions in the environment, including light, occupancy, temperature, noise level, and anything relevant to SE. One type of sensor is called a load sensor, which detects if a wire is currently conducting electricity to power an load such as an appliance or a light in active mode. One reason this is needed is that appliances such as some TV sets use a toggle-style on/off control without the ability to send separate turn-on or turn-off commands. A load sensor can provide the feedback necessary for the SE system to generate the right commands to properly control such appliances.

An actuator node contains an actuation device such as a motor for pulling a curtain, locking or unlocking a door, or turning on or off a switch. One type of actuator is a node for emitting an infrared (IR) remote-control commands. This is a simple, quick way to interface with legacy appliances and loads that are not equipped with our wireless module but already implements an IR remote-control interface. It can be used to control either an individual device or a set of devices. Another special type of actuator is a USB dongle that can act as a keyboard attached to a computer for generating a shut-down event to properly shut down a computer and powering it on demand, as most computers should not be shut down by simply cutting off its power.

3) Tags: A tag is any BLE device that is registered by the user as an identifier in our SE system. Tags can come in the form of "stickers" used for lost-item tracking and are often called *proximity tags*, which typically last for one year on a coin-cell battery. They are commonly used in conjunction with smartmobiles that act as *proximity sensors* that can scan and detect nodes in proximity. In addition, smartmobiles and any BLE device (e.g., fitness bracelets) can also act as proximity tags, as long as they implement the profile for proximity tags. We use tags as one of the possible mechanisms for localized access control.

B. Terminals and Gateway

A *terminal* is a user-interface device that can issue commands to the SE system and can render the status of the SE system. It can be a smartphone, tablet, a PC, and possibly a specialized embedded system with its own input devices and display. A terminal can connect with an SE node using an app or a client program. In fact, a terminal can connect either directly via BLE or indirectly over TCP/IP via the gateway to the SE nodes. A terminal is assumed to have Internet connectivity to a cloud server for authentication, data upload, and system administration purposes.

Moreover, once authenticated, a terminal can also play several roles in our SE system. First, it can also act as a node by providing sensing data just like any other sensor node. By registering a terminal such as a smartphone, it can also act as a proximity tag to identify its user for tag-based access control without having to carry an extra tag. Moreover, a terminal can also play the role of a backup gateway in case the main gateway fails temporarily.

A *gateway* is a protocol-bridging device between the Internet and the SE network. The gateway can sit on the local-area network (LAN) side of an access point. The gateway serves several purposes. First, it allows a remote terminal or a local

terminal without BLE to participate in the SE network by bridging over TCP/IP. Second, it can contain a local copy of the SE configuration and the access-control list, so that it can control access attempts by bridged devices such as a remote terminal or a locally bridged terminal. Third, a gateway helps with notification of status changes. This means if the change of state of an SE node (e.g., a light being turned on) should be propagated to another node or a terminal (e.g., a smartphone app shows that the light has been turned on), then one possible way is to push the event to the gateway, which in turn redistributes the events to all other subscribing devices on the SE network and the cloud.

The important point with our approach is that although the gateway can serve all these purposes, it is not an essential part of the operation. That is, it is possible for a terminal to send a command directly to a node without going through a gateway, and once the nodes are configured, they can perform access control on their own without relying on the gateway.

C. Cloud Backend

Our SE system can also work with a cloud backend. The cloud backend serves several purposes: authentication, support for access control, and configuration backup. First, it supports authentication of users on their terminal. Second, when accessing an SE system from a remote site, the terminal connects to the cloud that looks up the access control list as a way to enforce access control to the SE system. Third, the cloud stores a backup copy of the SE configurations, with the help of either the gateway or a terminal that has authenticated as a system administrator. The reason for backing up the configuration is that in case a node crashes and requires replacement, then the administrator can install a replacement node and restore the backed-up setting in it.

IV. System Features

This section describes the features provided by our SE system. We first describe the mechanisms for the nodes, terminals, and gateway to work together, including state propagation and conflict resolution. Second, we describe our two distributed access-control mechanisms, including role-based and tag-based ones.

A. Network Formation

The first step in setting up an SE system is network formation, somewhat analogous to forming a private Wi-Fi network using an access point and admitting additional nodes into it. In our case, a smartphone app is used for the initial network formation. It first defines a new private network and asks the user to set up the administrative account's password. BLE nodes and gateways capable of participating in the SE network are paired, and the app configures them with the proper network identifier and key.

The administrator can create users and roles for each device and write the information via our BLE profile to the nonvolatile memory of the device. The standard roles include administrator, host, guest, viewer, and others. More details about these roles are given in Section IV-C.

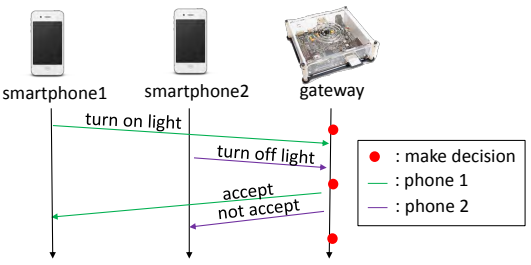

Fig. 2: Example of Conflict Resolution.

B. Multiple Access

By multiple access, we mean that a given device may be accessed by multiple other devices, including terminals (smartphones, tablets, PCs), the gateway, and other nodes such as sensors, timers, and macro buttons. There are two modes of multiple access: centralized and distributed.

1) Centralized Multiple Access: In centralized multiple access, all accesses go through the gateway, which can serve as the coordinator and arbitrator. This is how virtually all SE systems are done today. In this mode, the gateway acts as BLE master while all other devices in the network act as BLE slaves, including smartmobiles. The gateway receives all incoming commands and determines if they are in conflict (e.g., turning on and off a given device) at the same time. By "same time", we mean that commands are received within the given discrete time step. They can be resolved based on several policies: they are resolved primarily by priority order and secondarily by sequential order. At the end of each time step, a resolved command is then forwarded to the target device. All commanders that fail arbitration are sent a negative acknowledgment (NACK) by the gateway with a code that indicates failed arbitration. The command is forwarded to the target device, and a positive acknowledgment (ACK) from the target device is then forwarded to all winners in this round. The reason there can be multiple winners is that multiple nodes or terminals can all issue the consistent commands (e.g., all wishing to turn on a given device).

2) Distributed Multiple Access: In distributed multiple access, all commands are issued directly by the terminal or node to the target node, without going through the central gateway. In BLE, nodes can communicate with each other either in broadcast mode without pairing or as master-slave while paired. It is more natural for a smartmobile to act as a BLE master, but once it pairs with a slave device, it would preclude multiple access by another smartmobile or even other nodes such as sensors until the connection is broken. To enable multiple access in a distributed way, our smartmobile and target devices pair only briefly for a transaction and disconnects shortly after, so that it allows the possibility of other smartmobile to also pair with the same target device. We exploit BLE's feature to quickly re-pair them on demand, so that the users will perceive minimal latency. Conflict may happen in multiple access when the timer may try to turn off a light while an occupancy sensor is turning it on, or when one or more users may also try to change the light's state all at the same time. In this case, conflict resolution is done at the target node, which decides which commands to accept or

reject, without relying on centralized control.

3) Central Gateway vs. M2M: To enable node-to-node communication in our SE system, two modes of operation are supported: via the central gateway and direct M2M communication. One common example is an occupancy sensor that detects a period of no motion and sends a command to turn off a set of lights in the same room. In centralized mode, all nodes act as slaves to the gateway as the master in the network. This is similar to how a Wi-Fi network works. On the other hand, in M2M mode, a node switches between master and slave roles over time. All nodes normally remain unpaired but advertise themselves as available BLE slaves. A node that has a command to transmit (e.g., an occupancy sensor) switches to BLE-master role and pairs with the target light switch as a slave to transmit the command. After the transaction, they disconnect and the occupancy sensor switches back to an unpaired slave.

C. Role-Based Access Control

We adopt Role-Based Access Control (RBAC) [8] for defining the relationship as shown in Fig. 4. The definition consists of users, roles, permissions, and objects.

user	A *user* is a registered ID that is assigned one of more roles. Examples of users are mom, dad, girl, boy, and friend.
role	A *role* is a group ID that can be assigned to users. Examples of roles include family-adult (assigned to users mom and dad), family-child (assigned to users girl and boy), and family-guest (assigned to user friend).
object	An *object* in RBAC means a node or a sub-domain of nodes in our SE system. Examples of *coarse-grained* objects may be living room, kitchen, master bedroom, and child bedroom. It is possible to define finer-grained objects, such as "living-room lights," "DVD-player," "livingroom-TV," etc., which are all part of the "living room" coarse-grained object.
operation	An *operation* is a read or a write of an object's state. For example, for a fine-grained object such as a light switch, reading means to find if the light is on or off, and a write means to turn on or turn off the light.
permission	A *permission* is defined as a set of allowed (read, write) operations on an object. Examples of permissions include (living room, read), (master bedroom, read), (master bedroom, write), and other combinations.

The key of RBAC is that a user inherits the access authority from the role, and there is no need to define each user's access authority separately. The family-adult role may get assigned read and write permissions to all objects. The family-child role may get assigned read and write permissions to (nodes in) child bedroom and living room, plus read permission to kitchen. The family-guest role may get assigned read permission to living room only.

D. Tag-based Access Control

Tag-based access control means using proximity tags as the authentication mechanism instead of an explicit login to a cloud account. One scenario is when the user touches a physical button to turn on or off a light, the node attached to the physical button scans proximity tags [9] within its RF range and looks up their owner IDs. If a tag that belongs to an authorized user in proximity, then the action is permitted and executed. This effectively accomplishes similar access control as an RFID tag or an NFC card, except the user does not need to take out the tag, and any BLE device such as a smartphone or a fitness band can all be registered to serve as identifying tags, as long as they implement the proximity tag profile.

We further allow conjunctive tags in addition to disjunctive. That is, the security may be considered too weak by granting access as long as one authorized tag can be detected within range. To strengthen security, we can require two or more tags *conjunctively* to be granted access. This way, if a tag is stolen, the security is not compromised since one key alone is useless for the thief to gain access.

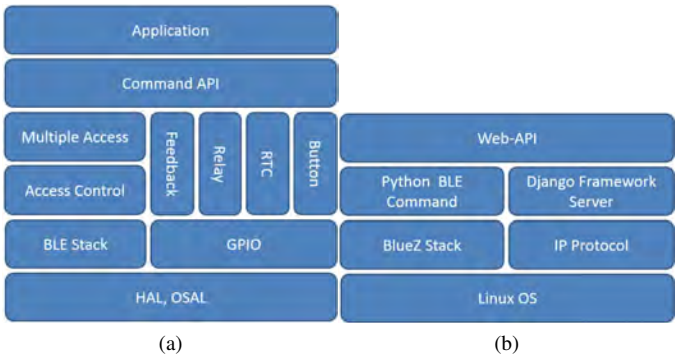

Fig. 3: (a) Firmware architecture on our BLE-based SE node. (b) Software architecture on the gateway.

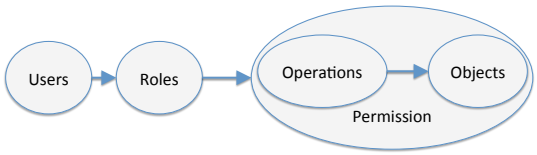

Fig. 4: Relationship of role-based access control

V. Implementation

This section describes a prototype of our proposed SE system. We use modules from a commercial vendor but replace their proprietary RF module with our BLE module. We describe the BLE-enabled SE node, gateway, smartphone app, and the cloud.

A. BLE-enabled SE Nodes

We prototyped our SE nodes based on commercial SE modules from TechCity [10]. Their modules include lighting modules, socket modules with load sensors, RF-to-IR hub, curtain motors, and others found in modern SE systems. One reason for our choice is its modular design: their RF modules are on a separate stacked PCB that can be easily removed and replaced with our own node using a modular connector, which also supplies power to our node. We adapt our previous

BLE node design to the form factor and connector required by the TechCity module so that they can be installed identically to the commercial products. Their power-control modules are specially designed to withstand very high instantaneous current, making them robust against transients commonly seen especially with fluorescent lights. Another robustness feature is that in addition to the wireless interface, the TechCity module also supports RS-485 wired interface as a backup in case the wireless communication fails. We attach our node to a touch panel to replace the existing traditional light switch on the wall.

As shown in Fig. 5, our BLE node is based on the TI CC2541 microcontroller unit (MCU) [11] with an integrated BLE transceiver. It has an 8051-compatible core with 8 KB SRAM and 256 KB code flash. Every node also includes an on-board real-time clock (RTC) chip, a serial flash, and a MicroSD slot for expansion storage if necessary. To be BLE compatible, we follow the firmware organization defined by TI, the operating system abstraction layer (OSAL), which is an event-triggered task dispatcher with some runtime support. It is required to work with the BLE stack provided by TI. The programmer can define the BLE profiles and write the application code to be compiled and linked with the BLE stack and OSAL to form the complete executable image. We also enable OAD (over-the-air download) feature for firmware update wirelessly.

Our BLE node can use its GPIO pins to sense the buttons when pressed, control the state of the switch, and to read the current state of the switch (i.e., whether it is currently on or off). Each BLE node has enough GPIO pins to control up to three buttons and three switches in one combo unit. This enables a user to turn on or off lights manually in the same way as traditional light switches. In other configurations, it is also possible to have one node controlling only buttons without switches as well as only switches without buttons.

The mode of operation can evolve over time. The system can start out just as a stand-alone node without joining any network. It can be paired with a BLE-enabled smartmobile using the BLE access code. To enable multiple access, after initial pairing, the node disconnects and re-pairs with the smartmobile only on demand. This enables all other smartmobiles to also access the same nodes as long as they have the shared passcode. While paired, the smartmobile app can also access the RTC on the node for timer control.

In more advanced mode, the user can increase the authentication and access-control strength. The administrative user sets the access control list that can be used distributed access control. One assumption that we make is that in case there is any change to the list (e.g., adding or revoking a user), the different nodes may be temporarily out of sync, but we expect the problem to be minor in that the changes are not immediate but must be staged first in advance. After all participating nodes have acknowledge their changes, then the update will take effect synchronously.

When a button is configured for access control mode, the node whose button is pressed will change its role to master to connect the tag and check the identity. In TI's implementation of BLE 4.0 stack, role switching from slave to master or from master to slave incurs 5 ms of latency, and this time is expected to be eliminated in the new BLE 4.1 standard, which allows a node to act as both a master and a slave at the same time. In any case, the total latency is dominated by the reading or writing of attributes.

Fig. 5: Light Controller with touch panel.

Finally, our node will continuously record the user data that happened when the device state changes. The node can know the current time from RTC and know the user from being accessed so it can store these data for future automating control via machine learning.

B. Gateway

We choose the PandaBoard ES as our gateway server, as shown in Fig. 6a. It is a consumer-grade single-board computer with a 1 GHz TI OMAP processor with dual-core ARM CortexA9, 3D graphics accelerator, and 1 GB stacked DDR2 RAM. The board contains a TiWi-BLE module that supports WiFi and BLE. We install Ubuntu 12.04 Gnu/Linux OS, BlueZ driver, Python, and Django framework in Pandaboard ES.

The gateway acts as a bridge that provides users with remote control ability. We use the Django framework to develop the web-API as an interface between smartmobile and gateway. To communicate with the BLE nodes, our gateway runs Python code to call the BLE driver API as provided by BlueZ. The flow is that after the gateway receives a request from the Internet, the gateway will send the request to the specific node via BLE and get response from the node, and then relay the node's message to the requester. The reason we choose Django framework to implement web-API is that it is the same Python runtime environment between Django and BLE, which enables directly packaging the BLE commands as a class for the Django framework to use.

C. Cloud

Strictly speaking, a cloud backend is not required in our distributed scheme. In gateway mode, only the gateway is required but not the cloud. Instead, the gateway contains a local copy of all the necessary functionality: authentication, access control, configuration backup, and web-API for remote access. Therefore, in the initial prototype, we excluded the cloud part to keep our evaluation more focused.

D. Smartmobile Terminal

The smartphone app provides a user interface for viewing, controlling, and configuring the SE nodes individually or in groups, including setting the RTC, schedule, and sensor-triggering preferences for each user. Note that the smartmobile

(a) (b)

Fig. 6: (a) Gateway (b) GUI.

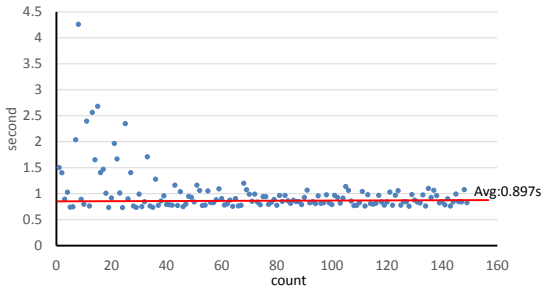

Fig. 7: Measured latency over 150 times.

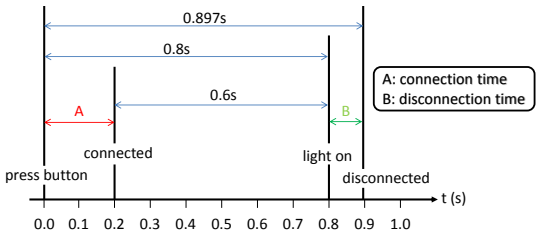

Fig. 8: Latency analysis.

can be used either as a local wireless control GUI or as an over-the-Internet remote control. For local wireless control, the smartmobile uses BLE protocol to communicate with the node directly without need for a gateway or infrastructure. Fig. 6b shows part of our app GUI for iOS 7. Only the administrator can configure the access control list and send it to the node. For remote control, users may wish that when they are out of home, they want to know and control the state of devices outside. To support this access modality, we design the gateway to provide web-API for commands from the smartmobiles.

VI. Evaluation

This section presents evaluation results from a number of experiments we conducted on our prototype SE system. We evaluate its performance in terms of the time latency and power overhead, and we discuss issues with access control.

A. Latency

Latency refers to the time difference from triggering an event on one node to the observable response on the target node. Specifically, we measure the latency from the time the user taps a button on the smartmobile app to the observable signal on the target node. The result is about 0.8 s. Note that this time includes access control overhead by authentication checking.

In more detail, we rely on programming to make the events observable for the purpose of time measurement. On the terminal side, we connect the smartmobile in development mode by running it in tethered mode to enable logging messages on the host MacOS X computer. We log the time when the button on the screen and when BLE connection is terminated.

Fig. 7 shows the measurement results. The average time of 0.897 s is obtained by taking the average of 150 measurements. On the node side, we program the GPIO pin to output a high value when the node is connected and output a low value when the node turns on the light. We test 20 times and the average time is 0.6 s. To further analyze the time consumed by different subsystems, Fig. 8 shows that

- connection takes $A = 0.8 - 0.6 = 0.2$ s, and
- disconnection takes $B = 0.897 - 0.8 = 0.097$ s.

The access control takes a trivial amount of time. With the default setting for the connection interval of 100-1000 ms, it takes 0.6 s from having established the connection to changing state of the switch. More specifically, it takes 460 ms, 80 ms, 60 ms, and 50 ms to receive the first packet, second, third, and fourth packet, respectively.

The difference between 0.8 s and 0.897 s is the time doing the disconnection. The latency that the user cares about is the time for turning on/off the light since pressing the button on the screen. Therefore, the time of doing disconnection is not contained in the latency. Also, the difference between 0.8 s and 0.6 s is the time for establishing the connection.

We set the connection interval (Section II-C) between 100 ms and 1000 ms. During a transaction, the smartmobile sends four packets to the target node for the user account, password, the on/off command, and the connection-termination command. Although the smartmobile can also unilaterally terminate the connection, in practice, it takes 2 seconds for iOS to disconnect, while the node can disconnect much more quickly. Therefore, for practical implementation, we decide to ask the node to disconnect rather than making the smartmobile disconnect.

B. Power Consumption

Power consumption is perhaps less of a concern for SE systems, because they can run on utility power instead of battery. However, such *vampire power* can add up to a significant amount of energy over time. We focus the power consumption on the node, which is the scope of our design, while the switch module can be improved separately by the vendor.

We divide power into idle mode and RF-active mode. Using power mode 2 on the CC2541 MCU, the voltage regulator to the digital core is turned off, while the registers and RAM contents are retained by the unregulated 2 V and 3.6 V power supplies and clocked by the 32 KHz oscillator. The system goes to active mode on reset, an external interrupt, or when the Sleep Timer expires.

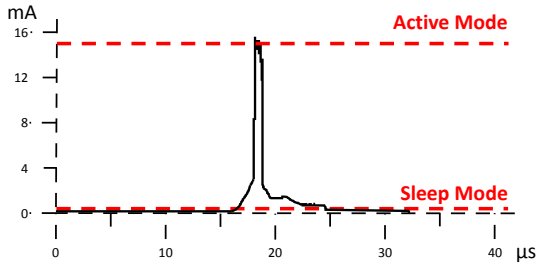

Fig. 9: Measured current consumption at 5 V.

Fig. 9 shows the measured current consumption of the node. It is measured by connecting a 1-Ω shunt resistor between the 5 V supply and the Vcc input to the node. During RF-active time, the node including MCU and RTC but excluding the TechCity module consumes about 14∼16 mA during advertising, 24 mA during connection, and 0.32 mA while idle. By changing to power mode 2, the current consumption drops to about 30 μA in sleep mode.

C. Discussion

1) Master-Slave Role Switching: Currently, the TI implementation of the BLE stack takes 5 ms for switching role. This is a short amount of time and is not noticeable for all practical purposes, although it must be programmed into the firmware explicitly. The new BLE 4.1 standard (such as that implemented by CSR) solves this problem by allowing a node to be both a slave and a master at the same time.

2) Replay Attacks: Most existing SE systems are vulnerable to *replay attacks* in that they merely transmit stateless, unencrypted command codes. An attacker can tune to the same frequency channel and record the signal that encodes the command, and at a later time, it can simply playback the signal. Note that replay attacks can work even if the command is encrypted, because the encrypted output of the same input and the same key will be identical, and the replay attacker does not even need to understand the way the command is encoded.

In BLE, attacks are much more difficult but not impossible, primarily during the initial pairing process [12]. Its weakest point is the key exchange protocol, since a 6-digit pin may not hold up to brute-force attacks, while OOB could be feasible by special hardware. Sniffers such as Ubertooth can be used for collecting packet data but not OOB. Also, BLE supports frequency hopping, which is not easy to crack, but if one snoops sufficiently long on a channel, then it is possible to determine the sequence. However, even then, BLE supports AES-128 hardware-accelerated encryption and decryption, as well as sending authenticated data over an unencrypted ATT bearer between two devices with a trusted relationship. The sender signs the data (command) with a signature composed of a Message Authentication Code generated by the signing algorithm and a counter. The counter is used to protect against a replay attack and is incremented on each signed Data PDU sent. One way to further strengthen the security is to also do encryption at the application level.

VII. Conclusion

We propose a new SE system that supports access control and multiple access to make them applicable to public settings. Unlike most SE systems that rely on centralized control, a distinguishing property with our system is that our system can operate by M2M communication without having to go through a gateway device, which can become a central point of failure. It supports several access control schemes: simple BLE access code, access control list, and proximity tags. BLE access code and proximity tags are inherently M2M schemes, while the access control list can be maintained on the cloud or on the gateway and be distributed to individual nodes. Our experimental results show that our proposed SE system can achieve reasonable latency and very low power overhead. We believe that our use of BLE is a promising choice in turning SE into a powerful IoT by leveraging direct M2M interaction over BLE to enable context-aware behavior that will make the lighting system a step closer to being smart, not just automating the control.

Acknowledgments

This project is sponsored by our funding agencies.

References

[1] N. Dou, Y. Mei, Z. Yanjuan, and Z. Yan, "The networking technology within smart home system - ZigBee technology," in *Proc. International Forum on Computer Science-Technology and Applications, 2009. IFCSTA '09.*, vol. 2, 2009, pp. 29–33.

[2] K. Gill, S.-H. Yang, F. Yao, and X. Lu, "A ZigBee-based home automation system," *IEEE Transactions on Consumer Electronics*, vol. 55, pp. 422–430, 2009.

[3] D.-M. Han and J.-H. Lim, "Design and implementation of smart home energy management systems based on ZigBee," *IEEE Transactions on Consumer Electronics*, vol. 56, pp. 1417–1425, 2010.

[4] K.-J. Lin, N. Reijers, Y.-C. Wang, C.-S. Shih, and J. Hsu, "Building smart M2M applications using the WuKong profile framework," in *Proc. IEEE International Conference on and IEEE Cyber, Physical and Social Computing,*, 2013, pp. 1175–1180.

[5] C. Gomez and J. Paradells, "Wireless home automation networks: A survey of architectures and technologies," *Communications Magazine, IEEE*, vol. 48, pp. 92–101, 2010.

[6] J. Kim, G. Boulos, J. Yackovich, T. Barth, C. Beckel, and D. Mosse, "Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes," in *Proceedings of the 2012 Eighth International Conference on Intelligent Environments*, ser. IE '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 206–213. [Online]. Available: http://dx.doi.org/10.1109/IE.2012.57

[7] Z. Wang and X. Xu, "Smart home M2M networks architecture," in *Proc. 2013 IEEE Ninth International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, 2013, pp. 294–299.

[8] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[9] K. Gautham, G. Raghav, V. Krishnamurthy, and N. Raajan, "Personnel security system using Bluetooth Low Energy (BLE) tag," *International Journal of Engineering and Technology (IJET)*, vol. 5, pp. 1527–1534, 2013.

[10] TechCity Technology Co., Ltd., "e^2·LiVE," http://www.techcity.com.tw, 2013.

[11] Texas Instruments, "CC2541 2.4GHz Bluetooth Low Energy System-on-Chip," http://www.ti.com/product/cc2541, 2013.

[12] M. Ryan and iSEC Partners, "Bluetooth: With low energy comes low security," in *Proc. USENIX*, 2013.