A Privacy Preserving Communication Protocol for IoT Applications in Smart Homes

Tianyi Song, Ruinian Li, Bo Mei, Jiguo Yu, Xiaoshuang Xing, and Xiuzhen Cheng, *Fellow, IEEE*

***Abstract*—The development of the Internet of Things has made extraordinary progress in recent years in both academic and industrial fields. There are quite a few smart home systems (SHSs) that have been developed by major companies to achieve home automation. However, the nature of smart homes inevitably raises security and privacy concerns. In this paper, we propose an improved energy-efficient, secure, and privacy-preserving communication protocol for the SHSs. In our proposed scheme, data transmissions within the SHS are secured by a symmetric encryption scheme with secret keys being generated by chaotic systems. Meanwhile, we incorporate message authentication codes to our scheme to guarantee data integrity and authenticity. We also provide detailed security analysis and performance evaluation in comparison with our previous work in terms of computational complexity, memory cost, and communication overhead.**

***Index Terms*—Chaotic systems, Internet of Things (IoT), privacy preservation, smart home systems (SHSs).**

I. Introduction

THE DEVELOPMENT of the Internet of Things (IoT) [1]–[7] has led to enormous IoT applications, such as intelligent transportation systems [8]–[10], smart shopping systems [4], and smart home systems (SHSs) [2], [11]. Among various IoT applications, the design of SHSs has drawn great attentions from both academic [2], [11] and industrial [12], [13] as the SHSs are closely related to people's lives.

However, existing architectures and designs of SHSs rarely take much consideration of the security and privacy issues. Consequently, in an SHS with low security strength, user's privacy information can be easily exposed to strangers or other malicious entities. For example, eavesdroppers can collect and aggregate the traffic information to profile a household.

Manuscript received January 29, 2017; revised April 17, 2017; accepted May 3, 2017. Date of publication May 23, 2017; date of current version December 11, 2017. This work was supported in part by the National Science Foundation under Grant CCF-1442642 and Grant CNS-1318872 and in part by the National Natural Science Foundation of China under Grant 61672321 and Grant 61373027. *(Corresponding author: Jiguo Yu.)*

T. Song, R. Li, B. Mei, and X. Cheng are with the Department of Computer Science, George Washington University, Washington, DC 20052 USA (e-mail: tianyi@gwu.edu; ruinian@gwu.edu; bomei@gwu.edu; cheng@gwu.edu).

J. Yu is with the School of Information Science and Engineering, Qufu Normal University, Rizhao 276826, China (e-mail: jiguoyu@sina.com).

X. Xing is with the School of Computer Science and Engineering, Changshu Institute of Technology, Suzhou 215500, China, and also with the Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou, China (e-mail: xiaoshuang_xing@163.com).

Digital Object Identifier 10.1109/JIOT.2017.2707489

Specifically, there are a few types of information that can be captured by simple eavesdropping attacks.

1) The end devices in a smart home probably send data more frequently to the central controller when user(s) is/are using them.
2) The types of the end devices being used can reveal the identity of the user in the house.
3) Smart meter reports to the utility company can exhibit certain patterns in the use of utilities, which enables the attackers to further identify the user in the house.

As a result, attackers can stay in the shadow and infer whether there is anybody in the house or who is in the house so that they can break into the house or cause life-threatening situations. Although SHSs and wireless sensor networks (WSNs) [14]–[17] are quite similar, SHSs have their unique features, such as fast adaptability to the environment, automatic adjustments in the behaviors of different components, etc. Therefore, SHSs can be treated as a special type of WSN like the body area networks [18]–[20].

In this paper, we investigate the state-of-art designs of SHSs and propose our own generic design of an SHS. Note that we presented a communication scheme for SHSs in [21], which can be treated as a preliminary version of this paper. Nevertheless, the communication protocol detailed in this paper is more efficient and possesses better secure and privacy-preserving features.

We outline two major challenges of designing a secure SHS as follows.

1) *Privacy*: Across the existing designs of SHSs [2], [11], [22], [23], the communication methods between any two components of an SHS are achieved via wireless networking. As a result, an attacker can simply eavesdrop on the traffic to infer daily activities of the residents [24], which endangers the legitimate users.
2) *Efficiency*: A general architecture of an SHS consists of RFID tags, sensors, actuators, and a central controller. The RFID tags, sensors, and actuators are well known for their limited computing capabilities, which make them not capable of carrying out complicated computing operations, not to mention executing asymmetric encryption algorithms.

Therefore, it is necessary to design such protocols that can guarantee secure and privacy-preserving communications within the SHS with little computational overhead. An effort tackling these challenges was made by Jie *et al.* [2], which proposed using certification authority (CA) to ensure the authenticity of a sensor. An entity is able to manipulate another

entity in the SHS only if its CA passes the privilege check. However, this design relies on a trusted CA center to assign a CA to a newly registered entity, and this scheme lays too much computational overhead on the sensors.

We propose a generic architecture for future SHS design that contains home appliances, RFID tags, sensors, a central controller, and user interfaces. To ensure security and privacy preservation, we incorporate a chaos-based cryptographic scheme and message authentication codes (MACs) for data transmissions.

The chaos-based cryptography has been widely adopted in securing communications. A chaotic system is extremely sensitive to the initial conditions, which means that the outputs of two chaotic systems with slightly different inputs can be significantly different. This butterfly effect feature makes chaos-based cryptography extremely suitable for key generations. Furthermore, in a chaos-based cryptographic system, even if one of the keys is recovered, the attacker cannot recover the other keys with negligible possibility. Therefore, we adopt one of the popular chaotic systems—the logistic map—to generate one-time symmetric keys to secure data transmissions.

To further ensure the security and privacy-preservation, we use MAC to guarantee data integrity and authentication. The main difference between MAC and digital signature is that MAC is based on symmetric cryptographic primitives while digital signatures are built on asymmetric cryptographic primitives. Given that the deployed RFID tags, sensors, and actuators have limited computing power and that performing asymmetric encryption and decryption bares too much computing tasks on the agents, we choose MAC over digital signature to protect data integrity and authentication.

Our contributions are listed as follows.

1) To the best of our knowledge, we are the first to outline a generic architecture design for future SHSs. We categorize the deployed entities in the SHSs based on their functionalities and operation types.
2) We design and improve our previous privacy-preserving communication protocol to achieve better security and privacy-preservation properties with little computational overhead and good scalability.

The rest of this paper is organized as follows. Section II surveys the state-of-art work of SHSs. Section III presents our generic architecture design of future SHSs. The system models and design goals are presented in Section IV. We briefly describe our previous scheme in Section V. Then in Section VI, we present our improved privacy-preserving communication scheme. Section VII shows the security analysis and performance evaluation of our proposed scheme. Finally, we draw our conclusions in Section VIII.

II. Related Work

Recent years have witnessed a lot of propositions for the future smart home architectures. Most of the works focus on giving details of hardware specifications, physical devices deployment, and new user interfaces developing.

Wang *et al.* [11] proposed an IoT-based appliance control system with detailed specifications of hardware configurations. Their proposed architecture enabled householders to remotely manipulate home appliances via the Internet. However, in their design, each of the appliances was associated with a designated controlling module, which made the realization of such an SHS harder and more expensive. Another interesting scheme, named AmI@Home [23], not only enabled remote control over the smart appliances but also implemented a collaborative scheme that required family members collaborating with each other to set up policies for appliance managements. In 2013, Yang *et al.* surveyed and interviewed 23 participants who had the Nest Thermostat [13] installed in their homes. They published their findings [3] on how the intelligent thermostat system affected people's living experience and the shortcomings of the Nest thermostat system. According to [3], the Nest Thermostat system was having trouble understanding the users' intent when the users manually adjusted the thermostat.

All these works focused on the functionalities of their proposed SHSs, leaving the security issues unresolved.

Different than [3], [11], and [23], Jie *et al.* [2] incorporated security measures to their proposed system that utilized RFID tags to uniquely identify an appliance. They proposed a five-layer architecture that contained resource layer, interface layer, agent layer, kernel layer, and user application layer. The kernel layer is in charge of authorization, data transportation, and interacting with the agent layer. Particularly, Jie *et al.* introduced the use of CA to enhance the security strength of their proposed system. They proposed a dual checking process to verify if an entity is qualified to manipulate another entity. For example, when an agent is invoked for certain operation, it will check the certificates of the entities that require this operation. If the certificates are qualified, the agent will send its certificate back to those entities to assure them that the invocation has been received. The agent will carry out the operation if and only if both CA verification processes are successful. The use of CA does guarantee the security strength of the smart home to some extent, yet the implementation of CA on the agents features high computational complexity that will result in low energy efficiency.

Like in many other fields, such as social media [25], [26], mobile computing [27]–[30], big data [31]–[34], power grid systems [35], etc., concerns about privacy preservation are also challenging researchers. Recently, Fernandes *et al.* studied the security issues from the user interface level of an SHS. Specifically, they studied the Samsung-owned smart things that consists of more than 500 applications (called SmartApps) to explore potential security breaches brought by the SmartApps, and they published their findings in [36]. According to their study, about 50% of the SmartApps are over-privileged, which means these apps are either granted more privileges than their functions imply or they are granted access that are not explicitly requested or used.

On the architecture level, Chakravorty *et al.* [37] presented a privacy-preserving scheme for multiple SHSs to submit their data to a central data analytics. The identities of the users in each smart home and the associated data are hashed and

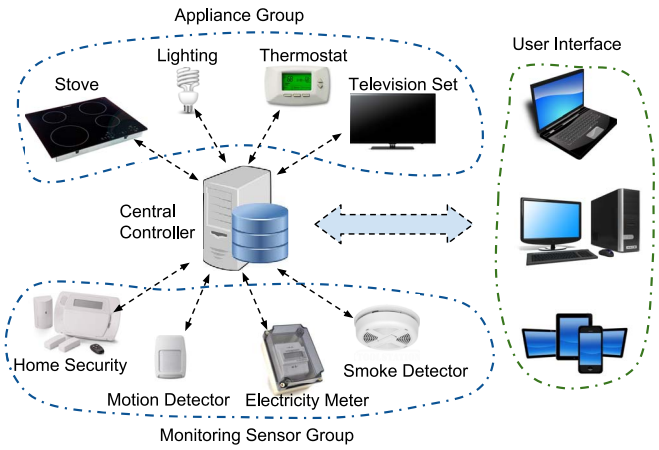

Fig. 1. Overview of the proposed smart home architecture.

stored in an identifier dictionary. When the data analytics was required to analyze the data, it de-identified the users and their data by looking them up in the identifier dictionary. By this way, it is guaranteed that the legitimate entities can acquire the real data and that malicious entities are not able to access the real data. Compared to [37], our proposed protocol seals the private data within the SHS where only authorized entities have access to it.

Chaos-based cryptography has been widely adopted in various applications in recent years [38]–[43]. Particularly, Hu *et al.* [42] employed chaos-based cryptography to securely outsource the matrix inverse computation to cloud/fog servers. The chaos-based cryptographic scheme was used to hide the original matrix in a new matrix produced by the keys generated by the chaotic system, and the server directly calculated the inverse of the matrix on the encrypted matrix and returned the results to the client.

III. Smart Home Design

In this section, we illustrate our design of an SHS from a system point of view. Our proposed architecture of the SHS consists of the following four groups of entities: 1) appliance group; 2) monitor group; 3) central controller; and 4) user interfaces as shown in Fig. 1. For simplicity, the entities that are in the appliance group and the monitor group are called agents. The agents communicate with the central controller via wireless networking technologies. The user controls the SHS through user interfaces.

Appliance group contains the home appliances including TV set, stove, oven, thermostat, etc. Each member of the group has a unique ID so that it can be uniquely identified by the central controller. The entities in the appliance group can perform limited operations, such as switching on/off, turning up/down, reporting status, etc.

Monitor group is formed by sensors and detectors, such as smoke sensor, motion sensor, electricity meter, and home security monitoring sensor. The sensors keep sensing the data and periodically send the data to the central controller. One can see that sending the data that reflects human activities to the central controller is risky as eavesdroppers can observe

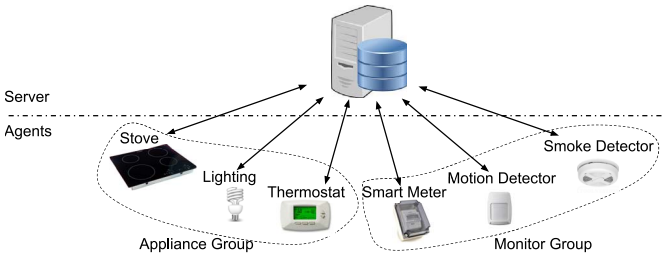

Fig. 2. Proposed system model.

and aggregate the traffic to infer private information, which motivate us to design a communication protocol to preserve the privacy of the household.

Central controller has two components: 1) a processor and 2) a database. The processor is capable of performing aggregation algorithms so that the controller can intelligently adjust the behaviors of the home appliances and sensors. The database stores the data reported by the sensors and appliances with time stamps for future reference. The database also stores the initial parameters of the logistic maps that are to be assigned to the sensors and appliances.

User interfaces offer a gate for the SHS to interact with users. The user interfaces take many forms such as a control pad installed in the house, website, softwares installed on personal computers, and applications on smart hand-held devices. The user interfaces enable the users to configure, monitor, and remotely control the SHS via the central controller.

Given the architecture of our proposed SHS, it is obvious that risks of privacy leakage is possible. Unsecured communications are vulnerable to malicious parties as private information can be inferred from the data traffic. In Section IV, we give the details of our designed communication protocol for agents to communicate with the central controller.

IV. Models and Design Goals

In this section, we present our system model and the threat model, as well as our design goals of the privacy-preserving scheme for the communications between the agents and the central controller.

A. System Model

In this paper, we consider a system model that has two components: 1) agents and 2) server, as shown in Fig. 2.

Agents: The agents are formed by the sensors, actuators, and monitors that are deployed over the smart home. They are activated as soon as their installations are finished, and they keep sensing contextual data in the house as time goes. They periodically report the collected data to the central controller. In the meantime, the agents continuously listen to responses and commands sent by the central controller.

Server: For simplicity, the central controller of the SHS is addressed as the server in this paper. As mentioned before, the server has a processor and a database. The server listens to the data reported by the agents and sends back responses

and commands. Meanwhile, the server is also an gateway for the users to configure, monitor, and control the SHS.

Note that we do not allow intercommunications between agents for the purpose of preventing collusion among the agents. Furthermore, with all the agents communicating only with the server, there is very little chance that a compromised agent is able to infect other agents without being detected by the server. However, the detection of compromised nodes are out of the scope of this paper.

In this paper, the problem we consider is privacy preservation and securing the communications between the agents and the server on the condition that the agents have restricted computing capabilities. In other word, we are proposing a scheme that: 1) keeps the data sent over the wireless links between agents and the server secret from a third party; 2) keeps the integrity of the transmitted data; and 3) preserves the privacy of the legitimate users.

B. Threat Model

We focus on the threats that come from the eavesdroppers under our model. Normally, an eavesdropper can easily observe the data sent over the air and extract the data that is collected by the agents if the SHS does not take into account the security and privacy issues. Therefore, the primary threat we consider is an attacker eavesdropping on the traffic data and aggregating the data to infer the private information about the household.

C. Design Goals

The design goals of our proposed model are twofold.

1) *Security and privacy preservation* need to be guaranteed. On one hand, the integrity of the transmitted data should be protected as the agents and the server will add an MAC to the original data to verify that the sender of the data transmission is who it claims to be and that the data is not tampered during the transmission. On the other hand, the transmissions are encrypted with secret keys shared by an agent and the server, which means that eavesdroppers cannot aggregate the data and obtain meaningful information with non-negligible possibility.
2) *Energy efficiency* is achieved while preserving the security and privacy of the household. The computational overhead on the agent's side is significantly reduced compared to that of adopting an asymmetric cryptographic system, which also avoids un-affordable delay on the agents' side when an urgent abnormality is detected.

V. Hash Function-Based Privacy Preserving Scheme

In this section, we briefly describe our hash function-based privacy preserving (HFPP) communication scheme presented in our preliminary work [21]. Note that HFPP works under the same system model and threat model as our chaos-based privacy preserving (CPP) scheme, and HPFF and CPP share the same design goals.

During initialization phase, the server takes the following steps.

1) The server generates secret keys k_i for all the agents for the purpose of generating the MAC. The server then stores the secret keys in a key table T_{key} in its database.
2) The server generates the initial secret keys $k_0^{(i)}$ for all the agents to perform encryption on messages. Again, the server stores the initial secret keys in a initial key table T_{init}.
3) When there is a new agent registered at the server, the server assigns one unique set of secret keys to the agent, with one of the keys used for MAC generation and the other for encryption.

Now, we illustrate each step of our HFPP scheme. We denote by A the batch of data that agent i is going to send at time stamp T. Each agent repeats collecting data and sending data to the server periodically.

A. HFPP on the Agents' Sides

Step 1: Agent i gathers its data to form collection A until the time for agent i to send A to the server comes. Agent i first checks the validity of its secret key value $k_j^{(i)}$. If $k_j^{(i)}$ has expired, i.e., $t_{\text{curr}} > \Delta t \times j$ where Δt is the life span of each secret key, and t_{curr} is the current system clock. Agent i updates its secret key to $h(k_j^{(i)})$, where $h(\cdot)$ is a one-way secure hash function.

Step 2: Agent i adds its secret key $k_j^{(i)}$ to the actual data as noise to form the ciphertext $C = (A||T) + k_j^{(i)}$, where T is the time stamp. Note that in HFPP, the agents perform addition operation as a way of encrypting the message.

Step 3: Agent i generates the MAC using its shared key k_i with the server. The MAC is denoted by MAC(C).

Step 4: Agent i concatenates C and MAC(C), and sends $C||\text{MAC}(C)$ to the server.

B. HFPP on the Server Side

Step 1: Upon receiving messages from agent i, the server obtains C and MAC(C) from the message and checks the validity of the time stamp T in C. If T is invalid, C discards this message.

Step 2: The server looks up the symmetric key k_i it shares with agent i in the key table T_{key}. Then, it computes the MAC of C using the shared key k_i with i, denoted by $\text{MAC}_{k_i}(C)$. Through comparing $\text{MAC}_{k_i}(C)$ and MAC(C), the server verifies the authenticity of the message. If $\text{MAC}_{k_i}(C)$ and MAC(C) do not match, the server discards the message and returns to step 1.

Step 3: Now that server has successfully verified the message, it can looks up the current secret key value $k_j^{(i)}$ in its initial key table T_{init} and use the current secret key to decrypt the message. In the meantime, the server checks if the key needs to be updated in the database.

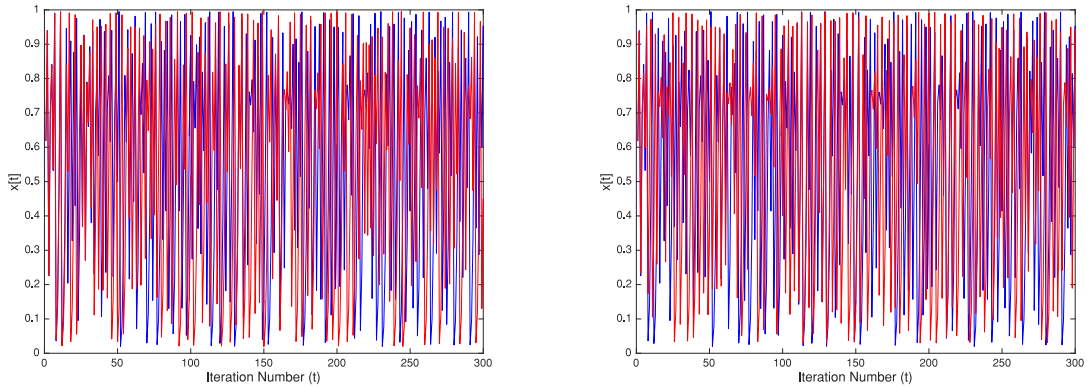

Fig. 3. Illustration of the chaotic behavior of the logistic map with three different sets of the initial values running 300 iterations: $x_0 = 0.6181$, $r = 3.98$ and $x_0 = 0.6182$, $r = 3.98$ are shown in the left subfigure, while $x_0 = 0.6181$, $r = 3.98$ and $x_0 = 0.6181mr = 3.97$ are shown in the right subfigure.

VI. Chaos-Based Privacy Preservation Scheme

In this section, we describe our CPP scheme for the agents in the SHS to communicate with the server. First, we briefly introduce the chaos-based cryptography. Second, we detail the four steps in our CPP scheme. Then, we describe the procedure of CPP.

A. Preliminary

Chaotic systems have been studied to achieve security goals for the past two decades. A chaotic system is characterized by its extreme sensitivity to the initial conditions and its topologically mixing property, both of which can guarantee the two principles of *confusion* and *diffusion*, when it comes to designing a crypto-system [43]. Logistic map is one of the most frequently used chaotic map for its simplicity [40], [42]. The logistic map is mathematically written as

$$x_{t+1} = rx_t(1 - x_t) \tag{1}$$

where $r \in (0, 4]$, $x_t \in [0, 1]$, and $t = 0, 1, \ldots, \infty$. The output of the logistic map exhibits chaotic behavior when $r > 3.56995$ as shown in Fig. 3.

In this paper, we adopt the logistic map to generate the secret key for each transmission between the smart home agents and the server as the computation is rather simpler than other chaotic systems such as the Lorenz system [41], the piecewise linear chaotic map [39], etc. The Lorenz chaotic oscillator is a continuous chaotic system; piecewise linear chaotic map is used more frequently in designing S-boxes.

B. Our Proposed Scheme

Our scheme contains the following four major steps.

1) *Key Generation:* The server generates a pair of secret keys for each agent and assign the pair of keys to each agent.
2) *Encryption:* The agents encrypt the data using the generated key.
3) *MAC Generation and Verification:* Both the server and the agents generate the MAC using their shared secret keys; and upon receiving a message from the server/agents, the agents/server check(s) the authenticity of the message using the same signing algorithm and the shared secret key.
4) *Decryption:* The server and the agents decrypt the message after the authenticity of the message is verified.

TABLE I
Parameters Table Stored in the Server

Agent Index	Map 1	Map 2	Counter
1	$(r_1^{(1)}, x_0^{(1)})$	$(r_2^{(1)}, x_0^{(1)'})$	0
2	$(r_1^{(2)}, x_0^{(2)})$	$(r_2^{(2)}, x_0^{(2)'})$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$

The reason for choosing symmetric encryption algorithms is that we want to reduce the energy consumption on the agents. However, key management and update issues are weighing down the security strength of symmetric encryption schemes. Therefore, we choose the logistic map to generate chaotic-look numbers as the one-time secret keys for secure transmissions. The details of each step is described as follows.

1) Key Generation: At the installation stage of an agent i, the server randomly chooses r_1, r_2, x_0, and x_0' such that $3.56995 < r_1 \neq r_2 < 4$ and $x_0 \neq x_0'$ as the parameters for two logistic maps $x_{t+1}^{(i)} = r_1 x_t^{(i)}(1 - x_t^{(i)})$ and $x_{t+1}^{(i)'} = r_2 x_t^{(i)'}(1 - x_t^{(i)'})$. The server shares (r_1, x_0) and (r_2, x_0') with agent i so that agent i is able to update the keys for the following data transmissions, which can be done when a new agent is added to the smart home network. For example, the agent may have a unique identifier and a corresponding passcode assigned by its manufacturer. Both the unique identifier and passcode must be submitted to the server in order to install the agent to the smart home network. After the installation, the passcode becomes obsolete. Only the server is able to communicate with the agent using the secret keys generated by its shared logistic map with the server. Meanwhile, the server stores the chosen parameters associated with agent i in the database shown in Table I for future reference. Once agent i starts functioning and is ready to send the collected data, it computes a pair of one-time secret keys $x_t^{(i)}$ and $x_t^{(i)'}$ from the two chosen logistic maps. The server does the same operation before it sends a message to agent i.

2) Encryption (On Agent i's Side): Agent i collects data for a period of time and stores the data in a vector $A = (a_1, a_2, \ldots)$.

Algorithm 1 Initialization

//Initialization
1: $t \leftarrow 0$
2: **for** each agent i **do**
3: The server selects two logistic maps: $x_{t+1}^{(i)} = r_1^{(i)} x_t^{(i)}(1 - x_t^{(i)})$ and $x_{t+1}^{(i)'} = r_2^{(i)} x_t^{(i)}(1 - x_t^{(i)'})$.
4: The server shares the parameters with agent i.
5: **end for**

Agent i appends the timestamp T and its index to the data to form the plaintext message $M = A||i||T$. The ciphertext $C = E_{x_t^{(i)'}}(M)$ is computed using a symmetric encryption algorithm.

On Server Side: The server encrypts the message along with the time stamp and computes the ciphertext $C = E_{x_t^{(i)'}}(M)$ using the chosen symmetric encryption algorithm.

3) MAC Generation and Verification (On Agent i's Side): Once the message is encrypted, the agent i computes the $\text{MAC}(C) = \text{MAC}_{x_t^{(i)}}(C)$ for the ciphertext and sends $C||\text{MAC}(C)$ to the server. When agent i receives a message from the server, it first checks the authenticity of the message by computing $\text{MAC}' = \text{MAC}_{x_t^{(i)}}(C)$ and compares it with the MAC carried by the message. Agent i proceeds to decrypt the message only if $\text{MAC}' = \text{MAC}(C)$.

On Server Side: The server computes $\text{MAC}(C) = \text{MAC}_{x_t^{(i)}}(C)$ using the secret key $x_t^{(i)}$ that is shared with agent i. The message sent out to agent i is $C||\text{MAC}(C)$. Upon receiving a message from agent i, the server first computes $\text{MAC}_{x_t^{(i)}}(C)$ and verifies $\text{MAC}(C)$ before it decrypts the message.

4) Decryption: Once the MAC is verified, the receiver of the message can decrypt the message with the shared key $x_t^{(i)'}$.

C. Privacy-Preserving Communication Protocol

Now, we describe the procedure of the proposed protocol. The corresponding steps of the protocol are shown in Algorithms 1 and 2.

At the initialization stage, the server picks a set of parameters for each agent (lines 2 through 4 in Algorithm 1) and stores the parameters in the database. At this stage, the round number t for computing the secret keys is set to 0. Once agent i is successfully deployed, it will notify the server.

Whenever agent i/server transmits a package to server/agent i, the sender generates the keys for encrypting the message and computing the MAC. The sender sends out a message that contains the encrypted data and the MAC of the data. The receiver, after receives a message from the other entity, computes the keys for decrypting and verifying the MAC. By doing this, we ensure the synchronization of the counter value on both the agents' sides and the server side.

VII. Security Analysis and Performance Evaluation

In this section, we first analyze the security strength of our proposed scheme in terms of data confidentiality, data integrity and authentication, and privacy. Then, we evaluate the performance of our proposed scheme from the following perspectives: computational complexity, communication overhead, and the security advantages of the proposed scheme compared to those of the HFPP scheme in [21].

Algorithm 2 Privacy-Preserving Communication Protocol

1: Agent i computes the symmetric key $x_t^{(i)'}$ from logistic map $x_{t+1}^{(i)'} = r_2^{(i)'} x_t^{(i)'}(1 - x_t^{(i)'})$.
2: Agent i generates ciphertext $C = E_{x_t^{(i)'}}(M)$.
3: Agent i computes the key $x_t^{(i)}$ from logistic map
4: $t \leftarrow t + 1$
$x_{t+1}^{(i)} = r_1^{(i)} x_t^{(i)}(1 - x_t^{(i)})$.
5: Agent i computes $MAC(C) = MAC_{x_t^{(i)}}(C)$.
6: Agent i sends $C||MAC(C)$ to the server.
7: Agent i updates its own counter $t \leftarrow t + 1$.
//Upon receiving a message $C||MAC(C)$ from agent i.
8: Server generates the pair of keys $(x_t^{(i)}, x_t^{(i)'})$ from the two logistic maps.
9: Server updates the parameter table with $t \leftarrow t + 1$.
10: Server updates the counter in the parameter table.
11: Server extracts the ciphertext C and computes $MAC_{x_t^{(i)}}(C)$.
12: **if** $MAC_{x_t^{(i)}}(C) = MAC(C)$ **then**
13: Server decrypts C using the key $x_t^{(i)'}$.
14: Server extracts the data A and timestamp T.
15: **if** T is consistent with the current system time **then**
16: Server responds the message based on the requests from the client;
17: **else**
18: Server discards the message.
19: **end if**
20: **else**
21: Server discards the message.
22: **end if**

As mentioned before, the deployed agents have restricted energy and computing power. Therefore, we choose to apply symmetric cryptographic schemes in the communication between the agents and server. For the sake of argument, we will use AES-256 symmetric encryption for the purpose of analyzing and evaluating the security properties and performance of our proposed scheme.

A. Security Analysis

Assume that the original message has n bits. We choose an HMAC with 160 bits to generate the MAC of a message. Now, we give our security analysis as follows.

1) Confidentiality: Since we adopt the AES cryptographic scheme with 256-bit keys, the probability of an eavesdropper successfully cracking the key using a brute-force manner is $(1/2^{256})$. Furthermore, the secret keys that are generated by logistic maps are only meant to be used for one-time encryption, which means that the key extracted by the attacker is of no use for decrypting any other messages. As a result, even if the attacker successfully cracks the secret key for one transmission, the attacker cannot crack all the other keys within non-negligible probability.

2) Data Integrity and Authentication: The data integrity is protected by tagging the MAC to the ciphertext. From the sender's point of view, MAC enables the sender to sign the message with its one-time secret key generated by the logistic map that is not known to a third party. From the receiver's perspective, the receiver can make sure that the message is not modified or tampered during the transmission by computing and verifying the MAC of the ciphertext. Thus, one can see that the data integrity is protected. In addition, a third party cannot pass the authentication by forging a fake MAC as the third party does not have knowledge of the shared secret key.

3) Privacy: Our proposed scheme can protect the users' privacy from two aspects. On one hand, we adopt symmetric encryption algorithms to encrypt the original data so that data is kept confidential from a third party. On the other hand, our SHS design requires the agents to periodically report data to the server, which makes it harder for an attacker to perform inference attacks by monitoring the traffic.

B. Performance Evaluation

Recall that in our previous HFPP scheme [21] the server assigns each agent a secret key for generating the MAC and an initial key for encryption. The agents computes the new secret key using a one-way hash function based on the old secret key. In CPP, the server generates two initial secret keys and two logistic maps for each agent. Each agent uses one of the initial key and the corresponding logistic map to generate the secret key for data encryption. The other initial key and the corresponding logistic map are used to generate the secret keys for computing the MAC.

1) Computational Complexity: In CPP, for one data transmission, the sender and the receiver need to conduct the following operations in a timely order: key generation, encryption, MAC generation and verification, and decryption. In particular, the key generation operation involves computing two secret keys from two different chaotic systems. One of the keys is used for encryption, and the other is used to generate the MAC tag. Additionally, the server takes one extra step of updating the counter in the database, which will take one Read/Write operation denoted as RW. We denote the complexity of each step as shown in Table II.

First, we analyze our scheme by comparing the computational complexity of the proposed scheme (CPP) and our previous scheme (HFPP).

On the Agents' Side: The computation complexity of the operations taking place on each agent is $G_k + C_E + G_M$.

On the Server Side: After the server receives the data, it performs the following computations: key generation, MAC verification, decryption, and counter update, which takes $G_k + V_M + C_D + RW$.

Although the computation complexity of HFPP and CPP are almost equivalent: in HFPP the key generation is based on one-way hash functions while in CPP the key generation is based on logistic maps, CPP achieves higher security level. In CPP, even an attacker successfully cracks one key, he is not able to compute the successive keys without knowing the specifics of the chaotic system.

TABLE II
COMPUTATION COMPLEXITY OF EACH STEP OF THE PROTOCOL

Key Gen	Encryption	Mac Gen	Mac Verification	Decryption
G_k	C_E	G_M	V_M	C_D

Second, we give a brief evaluation on the computational complexity of CPP compared to the computational complexity using public key cryptographic scheme. If traditional public key cryptosystem is utilized, then everytime an agent communicates with the server, it has to sign and encrypt the message, and correspondingly the server, once receives the message, has to decrypt the message using its private key and check the validity of the digital signature. Let R_e and R_d represent the encryption operation and decryption operation of public key system. The computational cost of signing the message and checking the digital signature are R_d and R_e separately. Each time an agent sends a message to the server, both the agent and the server have to compute $R_d + R_e$. In contrast, our protocol utilizes chaotic cryptosystem to achieve symmetric key generation and management, which makes symmetric cryptosystem easy to handle. In our protocol, each time an agent communicates with the server, it only needs to compute $C_E + G_M$, which has the same computational cost as $C_D + V_M$ on the server side. According to [44], the computational cost of C_E/C_D is much less than R_d, and the computational cost of G_M/V_M is much less than R_e. Therefore, CPP is much more efficient than a public key cryptographic system.

2) Memory Size: In this section, we evaluate the memory cost of our proposed scheme by following the protocol step by step as shown below.

1) The key generation stage outputs two 256-bit long secret keys for encrypting the data and generating an MAC.
2) The output of the AES-256 encryption is of the same length as the input. As mentioned before, the data size is n bits. Thus, the output of the encryption phase is n bits.
3) The generated MAC is 160 bit long.
4) To verify the MAC, the agent/server must take the cipher text and generate the MAC, which results in another 160 bit space.

Now, we calculate the detailed memory cost on both the agents' and the server's sides.

Agents: The memory cost is produced by storing two secret keys, the ciphertext, and the MAC, which is $256\times2+n+160 = n + 672$ bits. Note that the message that is sent out by one agent is $n + 160$ bits.

Server: After receiving a message, the server stores the message in its memory, which costs $n + 160$ bits. The server also has to generate two secret keys which will take $256 \times 2 = 512$ bits in the memory. To verify the MAC, the server needs to calculate $\text{MAC}_{x_i^{(i)}}(C)$, which will cost 160-bit space in the memory. In total, it takes $n + 160 + 512 + 160 = n + 832$ bits memory size of the server to process the received message.

Additionally, the server has a parameter table that stores the initial parameters of the logistic maps that are associated with the agents. We denote the number of agents as N. For each agent, there is an entry of data in the parameter table

that stores the agent's ID, and the initial parameters and the current counter for the logistic maps. Assume that the ID of an agent is 8 bit; the initial parameters, including r_1, r_2, x_0, x'_0, are all of 256 bit long; and the counter is 8-bit long. Thus, the storage space of one entry is $8 + 256 \times 4 + 8 = 1040$ bits; and thus the total space cost for the parameter table is $1040N$ bits.

3) Communication Overhead: In our proposed protocol, the key generation process does not introduce communication cost. Therefore, the communication overhead is produced during message exchanges between the agents and the server.

As introduced in Section VI, the message sent out by an agent is $C||\text{MAC}(C)$. The ciphertext C is n bits. The MAC is 160 bit long. Since we are using symmetric encryption, the messages exchanged between the agents and the server are encrypted and tagged with MAC; thus the total communication cost of one transmission is $n + 160$ bits.

4) Key Update and Management: We discuss how the chaos-based cryptographic system enables easy and secure key update and management from the following two aspects.

1) *One-Time Secret Key:* In [21], updating the key relies only on the computations of the chaotic systems shared between the agent and the controller instead of depending on both the one-way hash function and synchronized clocks. Meanwhile, the symmetric secret key is generated and used for one data transmission, which avoid vulnerabilities that are brought by repeatedly using the same key for a period of time. As a result, the one-time symmetric key achieves higher level of security strengths compared to the scheme.
2) *Low Storage Requirement:* In the HFPP protocol, for future reference, the central controller creates a database from the beginning to store all the secret keys that have been generated and used, as well as the time stamps of the keys being generated, which introduced huge demands of storage space. What is more, as time goes by, more and more records of keys and the corresponding time stamps will be added to the database, which makes it harder and slower for the central controller to query the keys that were used in the past. However, our CPP protocol requires no extra storage space for keeping records of the keys. The only reference to the old keys is the counter that counts the number of iterations that have been carried out, which makes the key management much easier.

VIII. Conclusion

In this paper, we first investigate the state-of-art designs of SHSs and then propose a generic smart home architecture for the future development of SHSs. Our proposed smart home architecture consists of an appliance group, a monitor group, a central controller, and user interfaces. In particular, the agents in the appliance group and the monitor groups periodically report the current statuses to the central controller. To achieve our security and privacy design goals, we propose a lightweight secure and privacy-preserving communication protocol that leverages chaos-based encryption and MACs. Considering the limited computing power on the agents deployed in the SHSs, we adopt a symmetric cryptographic system to encrypt the transmitted data. Additionally, we incorporate MAC to ensure the integrity and authenticity of the data. The one-time secret keys used for encryption and MAC calculation are generated based on two different chaotic systems. As a result, our proposed scheme achieves high efficiency and security level. Meanwhile, key management becomes easier and the requirement of memory size is reduced under our chaos-based cryptographic scheme.

References

[1] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, "Smart objects as building blocks for the Internet of Things," *IEEE Internet Comput.*, vol. 14, no. 1, pp. 44–51, Jan./Feb. 2010.

[2] Y. Jie, J. Y. Pei, L. Jun, G. Yun, and X. Wei, "Smart home system based on IoT technologies," in *Proc. 5th Int. Conf. Comput. Inf. Sci. (ICCIS)*, Jun. 2013, pp. 1789–1791.

[3] R. Yang and M. W. Newman, "Learning from a learning thermostat: Lessons for intelligent systems for the home," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. (UbiComp)*, Zürich, Switzerland, 2013, pp. 93–102. [Online]. Available: http://doi.acm.org/10.1145/2493432.2493489

[4] R. Li, T. Song, N. Capurso, J. Yu, and X. Cheng, "IoT applications on secure smart shopping," in *Proc. Int. Conf. Identification Inf. Knowl. Internet Things (IIKI)*, 2016.

[5] X. Zheng, Z. Cai, J. Yu, C. Wang, and Y. Li, "Follow but no track: Privacy preserved profile publishing in cyber-physical social systems," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2017.2679483.

[6] L. Qi, P. Dai, J. Yu, Z. Zhou, and Y. Xu, "'Time–location–frequency' aware Internet of Things service selection based on historical records," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 1, 2017, Art. no. 1550147716688696.

[7] J. Lin *et al.*, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, to be published. [Online]. Available: http://doi.acm.org/10.1109/JIOT.2017.2683200

[8] W. He, G. Yan, and L. Da Xu, "Developing vehicular data cloud services in the IoT environment," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1587–1595, May 2014.

[9] T. Song *et al.*, "Enhancing GPS with lane-level navigation to facilitate highway driving," *IEEE Trans. Veh. Technol.*, to be published, doi: 10.1109/TVT.2017.2661316.

[10] J. Lin *et al.*, "A novel dynamic en-route decision real-time route guidance scheme in intelligent transportation systems," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Columbus, OH, USA, Jun. 2015, pp. 61–72.

[11] M. Wang, G. Zhang, C. Zhang, J. Zhang, and C. Li, "An IoT-based appliance control system for smart homes," in *Proc. 4th Int. Conf. Intell. Control Inf. Process. (ICICIP)*, Beijing, China, Jun. 2013, pp. 744–747.

[12] (2016). *Akerman Smart Home Security*. [Online]. Available: https://www.ackermansecurity.com/

[13] *Nest Thermostat*. Accessed on May 1, 2016. [Online]. Available: https://nest.com/thermostat/meet-nest-thermostat/

[14] S. Cheng, Z. Cai, and J. Li, "Curve query processing in wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 11, pp. 5198–5209, Nov. 2015.

[15] S. Cheng, Z. Cai, J. Li, and H. Gao, "Extracting kernel dataset from big sensory data in wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 4, pp. 813–827, Apr. 2017.

[16] S. Cheng, Z. Cai, J. Li, and X. Fang, "Drawing dominant dataset from big sensory data in wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, 2015, pp. 531–539.

[17] B. Huang, J. Yu, D. Yu, and C. Ma, "SINR based maximum link scheduling with uniform power in wireless sensor networks," *KSII Trans. Internet Inf. Syst.*, vol. 8, no. 11, pp. 4050–4067, 2014.

[18] C. Hu, H. Li, Y. Huo, T. Xiang, and X. Liao, "Secure and efficient data communication protocol for wireless body area networks," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 2, pp. 94–107, Apr./Jun. 2016.

[19] C. Hu, N. Zhang, H. Li, X. Cheng, and X. Liao, "Body area network security: A fuzzy attribute-based signcryption scheme," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 9, pp. 37–46, Sep. 2013.
[20] C. Hu *et al.*, "OPFKA: Secure and efficient ordered-physiological-feature-based key agreement for wireless body area networks," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 2274–2282.
[21] T. Song, R. Li, X. Xing, J. Yu, and X. Cheng, "A privacy preserving communication protocol for IoT applications in smart homes," in *Proc. Int. Conf. Identification Inf. Knowl. Internet Things (IIKI)*, Wuhan, China, 2016.
[22] M. Naglic and A. Souvent, "Concept of smarthome and smartgrids integration," in *Proc. 4th Int. Youth Conf. Energy (IYCE)*, Siófok, Hungary, Jun. 2013, pp. 1–5.
[23] D. Fogli, R. Lanzilotti, A. Piccinno, and P. Tosi, "AmI@Home: A game-based collaborative system for smart home configuration," in *Proc. Int. Working Conf. Adv. Vis. Interfaces (AVI)*, Bari, Italy, 2016, pp. 308–309. [Online]. Available: http://doi.acm.org/10.1145/2909132.2926083
[24] H. Li, Y. He, L. Sun, X. Cheng, and J. Yu, "Side-channel information leakage of encrypted video stream in video surveillance systems," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, San Francisco, CA, USA, 2016, pp. 2617–2626.
[25] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Trans. Depend. Secure Comput.*, to be published, doi: 10.1109/TDSC.2016.2613521.
[26] Z. He *et al.*, "An energy efficient privacy-preserving content sharing scheme in mobile social networks," *Pers. Ubiquitous Comput.*, vol. 20, no. 5, pp. 833–846, Oct. 2016. [Online]. Available: http://dx.doi.org/10.1007/s00779-016-0952-6
[27] L. Zhang, Z. Cai, and X. Wang, "FakeMask: A novel privacy preserving approach for smartphones," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 335–348, Jun. 2016.
[28] Y. Wang *et al.*, "An incentive mechanism with privacy protection in mobile crowdsourcing systems," *Comput. Netw.*, vol. 102, pp. 157–171, Jun. 2016.
[29] C. Hu *et al.*, "Efficient privacy-preserving schemes for dot-product computation in mobile computing," in *Proc. 1st ACM Workshop Privacy Aware Mobile Comput.*, 2016, pp. 51–59.
[30] C. Hu *et al.*, "An attribute-based signcryption scheme to secure attribute-defined multicast communications," in *Proc. SecureComm*, Dallas, TX, USA, 2015, pp. 418–437.
[31] X. Jin, M. Zhang, N. Zhang, and G. Das, "Versatile publishing for privacy preservation," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, Washington, DC, USA, 2010, pp. 353–362. [Online]. Available: http://doi.acm.org/10.1145/1835804.1835851
[32] A. Dasgupta, N. Zhang, G. Das, and S. Chaudhuri, "Privacy preservation of aggregates in hidden databases: Why and how?" in *Proc. ACM SIGMOD Int. Conf. Manag. Data (SIGMOD)*, Providence, RI, USA, 2009, pp. 153–164. [Online]. Available: http://doi.acm.org/10.1145/1559845.1559863
[33] N. Zhang and W. Zhao, "Privacy-preserving data mining systems," *Computer*, vol. 40, no. 4, pp. 52–58, Apr. 2007.
[34] C. Hu *et al.*, "A secure and verifiable access control scheme for big data storage in clouds," *IEEE Trans. Big data*, to be published, doi: 10.1109/TBDATA.2016.2621106.
[35] Q. Yang *et al.*, "On false data-injection attacks against power system state estimation: Modeling and countermeasures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 717–729, Mar. 2014.
[36] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Proc. IEEE Symp. Security Privacy (SP)*, San Jose, CA, USA, May 2016, pp. 636–654.
[37] A. Chakravorty, T. Wlodarczyk, and C. Rong, "Privacy preserving data analytics for smart homes," in *Proc. IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, May 2013, pp. 23–27.
[38] W. S. Sayed, A. G. Radwan, and H. A. H. Fahmy, "Design of a generalized bidirectional tent map suitable for encryption applications," in *Proc. 11th Int. Comput. Eng. Conf. (ICENCO)*, Cairo, Egypt, Dec. 2015, pp. 207–211.
[39] P. Zhen, G. Zhao, L. Min, and X. Li, "A survey of chaos-based cryptography," in *Proc. 9th Int. Conf. P2P Parallel Grid Cloud Internet Comput.*, Guangdong, China, Nov. 2014, pp. 237–244.
[40] T. S. Chaware and B. K. Mishra, "Secure communication using TPC and chaotic encryption," in *Proc. Int. Conf. Inf. Process. (ICIP)*, Pune, India, Dec. 2015, pp. 615–620.
[41] P. Tobin, L. Tobin, M. M. Keever, and J. Blackledge, "Chaos-based cryptography for cloud computing," in *Proc. 27th Irish Signals Syst. Conf. (ISSC)*, Londonderry, U.K., Jun. 2016, pp. 1–6.
[42] C. Hu, A. Alhothaily, A. Alrawais, X. Cheng, and C. Sturtivant, "A secure and verifiable outsourcing scheme for matrix inverse computation," in *Proc. IEEE INFOCOM*, 2017.
[43] L. Kocarev, "Chaos-based cryptography: A brief overview," *IEEE Circuits Syst. Mag.*, vol. 1, no. 3, pp. 6–21, Third Quarter, 2001.
[44] *Crypto++ 5.6.0 Benchmarks*. Accessed on Nov. 1, 2016. [Online]. Available: https://www.cryptopp.com/benchmarks.html

Tianyi Song received the B.S. degree from the Department of Computer Science and Technology, Beijing Forestry University, Beijing, China, in 2011. She is currently pursuing the Ph.D. degree at the Department of Computer Science, George Washington University, Washington, DC, USA.

Her current research interests include secure and privacy-aware computing for the Internet of Things, security and privacy in cyber-physical systems, mobile computing, and wireless networking.

Ruinian Li received the B.S. degree in software engineering from Nanchang University, Nanchang, China, in 2011, and the M.S. degree in computer science from Suny Polytechnic Institute, Utica, NY, USA, in 2013. He is currently pursuing the Ph.D. degree with the Department of Computer Science, George Washington University, Washington, DC, USA.

He attended the joint program of Nanchang University and Suny Polytechnic Institute. His current research interests include network security, applied cryptography, and privacy preserving computations.

Bo Mei received the B.S. degree in material science and engineering from the Beijing Institute of Technology, Beijing, China, in 2010, the M.S. degree in interdisciplinary engineering from Purdue University, West Lafayette, IN, USA, in 2011, and the M.S. degree in computer science from George Washington University, Washington, DC, USA, in 2013, where he is currently pursuing the Ph.D. degree.

He began his research focusing on mobile computing in 2014. He has conducted extensive study on system applications of IoT devices and has published several papers as first author. His current research interests include the broad areas of system security of IoT devices and machine learning.

Jiguo Yu received the Ph.D. degree from the School of Mathematics, Shandong University, Jinan, China, in 2004.

Since 2007, he has been a Professor with the School of Computer Science, Qufu Normal University, Jining, China, where he is currently a Professor with the School of Information Science and Engineering. His current research interests include wireless networks, distributed algorithms, peer-to-peer computing, and graph theory. In particular, he is interested in designing and analyzing algorithms for many computationally hard problems in networks.

Dr. Yu is a Senior Member of the China Computer Federation.

Xiaoshuang Xing received the B.A. degree from North China Electric Power University, Beijing, China, in 2010, and the Ph.D. degree from Beijing Jiaotong University, Beijing, in 2014.

She is an Associate Professor with the School of Computer Science and Engineering, Changshu Institute of Technology, Suzhou, China. Her primary research interests include spectrum prediction, cognitive radio networks, Internet of Things, and mobile social networks.

Xiuzhen Cheng (F'15) received the M.S. and Ph.D. degrees in computer science from the University of Minnesota, Minneapolis, MN, USA, in 2000 and 2002, respectively.

She is a Professor with the Department of Computer Science, George Washington University, Washington, DC, USA. Her current research interests include privacy-aware computing, wireless and mobile security, cyber physical systems, mobile computing, and algorithm design and analysis.

Dr. Cheng is a member of the ACM. She was a recipient of the NSF CAREER Award in 2004. She has served on the editorial boards of several technical journals and the technical program committees of various professional conferences/workshops. She also has chaired several international conferences. She was a Program Director for the U.S. National Science Foundation in 2006 (full time) for six months, and from 2008 to 2010 (part time).