Wireless Personal Communications
https://doi.org/10.1007/s11277-018-6035-4

Lightweight Security Scheme for Internet of Things

Ahmed Aziz[1,2] · **Karan Singh**[1]

Abstract
The compressive sensing method presents itself as a promising technique in many fields specially for the Internet of Things and Wireless sensor networks applications. That is because, the compressive sensing has the major advantage of performing lightweight encryption and compression simultaneously. It leads to secure the network in addition to prolong the network life time. However, chosen plaintext attacks and key distribution are still major challenges facing the compressive sensing method. This paper focuses on the compressive sensing method according to security issue, and propose an efficient lightweight security scheme that addressee the previous challenges. Moreover, we use experimental data collected from a real sensors located in Intel Berkeley Research Lab.

Keywords Wireless sensor networks · Internet of Things · Compressive sensing · Security

1 Introduction

The Internet of Things (IoT) provides networking to connect people, things, applications, and data through the Internet to enable remote control, management, and interactive integrated services. Based on IoT, now everything is going to be connected. In addition, predictions are made that there will be 50 billion 'things' connected to the Internet by 2020. So, Internet of Things study is very important.

Actually, Wireless sensor networks (WSNs) is considered the most important element in the IoT model. Utilization of wireless sensors devices and other IoT technologies in green applications and environmental conservation are one of the most promising market segments in the future [1, 2]. The main task for IoT and WSNs sensors nodes is to sense the data then send them to the base station (BS). Therefore, security and energy efficiency are of major importance issues in IoT. As a significant number of applications (e.g. e-health) are based on resource-constrained wireless sensors that often convey sensitive and private

✉ Karan Singh
karan@mail.jnu.ac.in

Ahmed Aziz
ahmed.aziz@fci.bu.edu.eg

1 School of Computer and System Science, Jawaharlal Nehru University, New Delhi 110067, India

2 Benha University, Benha, Egypt

Published online: 26 October 2018

information [1]. Encrypt the transmitted data between sensors and the BS usually makes the security possible.

There are two types of encryption algorithms which are known as asymmetric and symmetric key algorithms. The data encryption algorithms based on asymmetric key like RSA and ECC [3] achieve high level of security. But they are not preferable for IoT and WSNs devices because, these devices have limited resources, in terms of processing power, memory, and storage. On the other hand, the data encryption algorithms based on symmetric key like DES and AES [4], although they don't require huge computational power and storage, but key exchange scheme is still required or keys have to be pre-stored. So sensors can be easily compromised when placed in outdoor environments [5]. Another interested major in IoT systems is energy efficiency. All most of IoT devices suffer from energy constrain problem. This energy is mostly spent during the sensors communication over its wireless radio interface. So prolonging the network life is also a major target issue.

In this paper, we address all of these issues by proposing lightweight security scheme which depends on compressive sensing (CS) [6] method. In CS, the signal can be successfully sampled less than the rate of Nyquist theory, if the signal is sparsed by natural or by transformer. In CS, the signal is sampled and compressed simultaneously, rather than sample it then compress like the other traditional compression techniques. The signal can be reconstructed again without any significant losing in the information. Also, CS has the major advantage of performing lightweight encryption and compression simultaneously, so energy-efficiency becomes possible.

The rest of the paper is organized as follows: The related work is briefly reviewed in Sect. 2. Section 3 presents Compressive Sensing background. Section 4, introduces the proposed approach to solve the presented problems. In Sect. 5, an example scenario is provided. In Sect. 6, we present a simulation of our approach. Section 7 is the conclusion to the proposed research paper.

2 Related Work

During the last few years, IoT applications such as e-health and transportation applications have attracted number of researchers. Security and energy efficiency are still big challenges face the IoT. To address these challenges several related work with significant contributions have been proposed.

The use of CS for security issue was firstly presented in [7], in which the authors suggested that the compressed samples obtained from random linear projections can be considered as ciphertext. Since the attacker would not be able to decrypt it unless the attacker knows the measurement matrix which used to compress and encrypt them. In this way, the entire CS scheme can be considered as a variant of the stream cipher

In [8], Marco et al. used CS to solve the authentication and tampering identification problems. In [9, 10] the Received-Signal Strength is considered for key generation, but these proposed techniques generate keys that are suitable for conventional encryption algorithms (e.g. RSA, ECC, etc). The works in [5, 11, 12] considered key generation using channel measurements for CS-based encryption without any key distribution scheme. These works, however have disadvantage for use in a IoT. They used too many steps to generate the key in the resource-constrained sensors side which lead to increase the power consumption. On the contrary, the proposed scheme shifts all mathematics computations to the BS side during the key generation and exchange steps.

Rachlin and Baron [13] show that if the attackers use incorrect encryption matrix to decrypt the data, then the the original data sparsity is lower than that of the decrypted data. In [14], Cambareri et al. shows that CS is not perfectly secure by performing a statistical analysis without focusing on the computational feasibility. In [15], Yu et al. used chaotic sequences to generate the measurement matrix. The paper [13] shows that, the measurement matrix provides secure computations from some attackers, such as brute-force attack and Ciphertext Only Attack (COA). The above-mentioned CS-based encryption algorithms retain computational secrecy only under some attacks, i.e., brute-force attack and ciphertext only attack. But these algorithms didn't take in consideration the Chosen Plaintext Attack (CPA) scenario. The first work focus on the CPA attacks was in [16], in which the authors create a secret sparsifying basis known as Fractional Fourier Transform. But this method is too difficult to be used in sensors which have constrain in storage and power. In [17] the authors also addressed the CPA attack by using chaotic sequences as a secret values that are processing and memory efficient. However, they did not mention how the legitimate users can exchange this secrets values in a secure way.

2.1 Our Contribution

In this paper, we propose an efficient lightweight security scheme (LSS) which is based on CS method. LSS aims to solve the security and energy efficiency issues for IoT. The highlight of our contributions can be listed list as the following:

- For security improvement, LSS uses simple scenario to generate and exchange the key. This key is shared between the BS and the sensor nodes. In which, the sensor nodes generates random number by using simple chaotic map [17]. Also, the BS generates two random numbers one of them is called seed which considered as the key by using 2-D chaotic maps [18]. After that, the both sides (sensor nodes and the BS) use the proposed key exchange algorithm to exchange these numbers simply and securely. This scenario applies only once before any transmission by the both sides.
- In order to enhance the security issue in terms of resisting against the chosen-plaintext attack, LSS presents an algorithm called data compression with encryption algorithm. In which, the sensor nodes generate a secret compressed samples by using secret value.
- LSS performs lightweight encryption and compression simultaneously, so it reduces the energy consumption and prolongs the network life time.

3 Compressive Sensing Background

The CS provides a direct method in which the data is compressed and sampled in one step in stead of sampling and then compressing such as conventional compression [6]. In addition, the CS reconstruction algorithm can successfully reconstruct the original data from the compressed samples without any prior knowledge [6]. The used notations through this paper are given in Table 1.

3.1 Mathematical Definition

Let $x[n], n = 1, 2, \ldots, N$, the collected set of sensors readings is vector in R^N,where N represent number of sensors. Any signal in R^N can be represented in terms of a basis

Table 1 Notions description

Notation	Description
x	Sensors readings
Ψ	Transform matrix
Φ	Measurement matrix
Θ	$M \times N$ matrix such that $\Theta = \Phi\Psi$
y	Measurement vector (compressed samples)
g	Sparse presentation of x
S	Sparse level (number of non zeros values)
$g_S \& e_1$	Two random numbers generated by the BS, such that g_S is considered as the seed
e_2	Random number generated by the sensor node
S_v	Secret value
y'	Secret compressed samples

of $N \times 1$ vectors $\{\Psi_i\}_{i=1}^{N}$. For simplicity, assume that the basis is orthonormal. Using the $N \times N$ basis matrix $\Psi = [\Psi_1|\Psi_2|\Psi_3|\cdots|\Psi_N]$ with the vectors Ψ_i as columns, a signal x can be expressed as [6]

$$x = \sum_{i=1}^{N} g_i \Psi_i \quad or \quad x = \Psi g \tag{1}$$

where g is the $N \times 1$ sparse presentation of x. The transform matrix $\Psi \in N \times N$ is an orthonormal basis. CS will focus on signals that have a sparse representation, i.e x has just S basis vectors , with $S << N$. That is, $(N - S)$ are zero and only S of g are nonzero.

By using the Eq. (1), the compressed samples y can be obtained from Eq. (2):

$$y = \Phi x = \Phi \Psi g = \Theta g, \tag{2}$$

where the compressed samples vector is $y \in R^M$,with $M << N$ and Θ is an $M \times N$ matrix.

3.2 Selecting Suitable CS Matrix

CS technique faces the problem of finding a compression matrix Θ which must allows the reconstruction of the original signal x of length-N from the compressed signal y of length M where $M < N$ and since $M < N$, the solution becomes ill-posed in general. To solve this problem, the measurement matrix must satisfy the following conditions:

1. Restricted Isometry Property (RIP) [19]: The salient feature of CS that it focuses on the spares signal in which the measurement vector y is just a linear combination of the S columns of Θ whose corresponding $g_i \neq 0$ (see Fig. 1). Now, CS changes the problem from recovering N from M to recover S from M, where $S << M$, which make the solution well-posed. The prime perquisite of the matrix Θ to ensure that the solution will be well-posed must satisfy the RIP condition:

$$1 - \epsilon \leq \frac{\| \Theta v \|_2}{\| v \|_2} \leq 1 + \epsilon \tag{3}$$

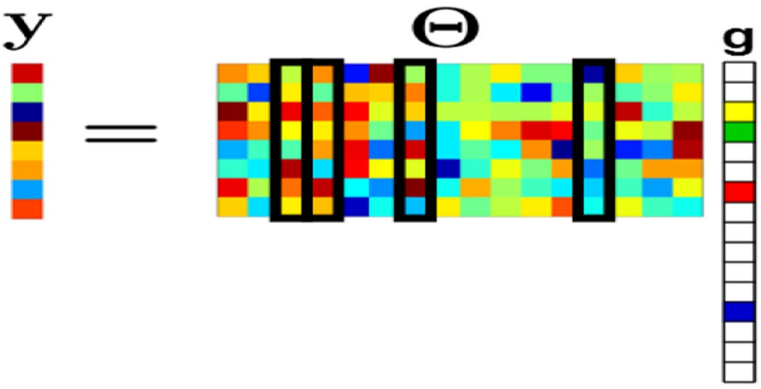

Fig. 1 Columns which create the measurement vector y [20]

for some $\epsilon > 0$ and any vector v sharing the same G nonzero entries as g. In words, the matrix Θ must preserve the lengths of these particular S-sparse vectors. In fact, it is very difficult to locate the nonzero values in s. Fortunately, to maintain a stable inverse for both S-sparse and compressible signals, only one condition is needed which is Θ to satisfy Eq. (3).

2. Incoherent: The matrix ϕ must be incoherence with the matrix Ψ. Therefor, the acquisition domain ϕ and the sparse domain Ψ must be almost uncorrelated.

Mostly CS methods use random matrices like Gaussian or Bernoulli distribution matrix, which fulfil all the previous conditions.

3.3 Signal Reconstruction

For many years, the problem of finding the solution to an unspecified set of linear equations has grabbed the attention in the literature. Different practical applications has been carried out for this problem, for example compressed sensing. In CS scenario, few measurements coefficients are available and the task is to recover a larger measurements. The signals are reconstructed from this incomplete set of measurements, depending on the fact that the signal has sparse representation. One of the simplest solutions to recover such a vector from its measurements Eq. (2) is to solve $\| L \|_0$ minimization problem that counts the number of non-zeros entries, the reconstruction problem turns to be:

$$x = arg\ min\|x\|_0 \quad subject\ to\ y = \Phi x \tag{4}$$

Thus, this $\| L \|_0$ minimization problem works perfectly theoretically. However, it is computationally NP-Hard in general [21]. It is computationally intractable to solve Eq. (4) for any matrix and vector. Fortunately, in the framework of CS, there are two families can be alternatively used to solve Eq. (2) with computationally efficient for this computationally NP-Hard problem. One is the basic pursuit that is a convex relaxation leading to $\| L \|_1$ norm minimization [22] and the other is greedy pursuit such as Orthogonal Matching Pursuit (OMP) [23], Stagewise Orthogonal Matching Pursuit (StOMP) [24] and Regularized Orthogonal Matching Pursuit (ROMP) [25].

4 Lightweight CS Security Scheme

Section 3 explains the simplicity of CS scheme in reducing the data dimension without go through a lot of complex mathematical computation, makes it a preferable method to compress the data traffic transmitted through WSNs and IoT. Another attractive characteristic of CS is that it can used to do both of data encryption and compression in the same platform. Observe in Eq. (2) that the original data x is converted to the compressed sample y by using matrix $\boldsymbol{\Phi}$. This operation is very similar to block ciphers encryption one, where x is the plaintext, y is considered the ciphertext and the matrix $\boldsymbol{\Phi}$ is the key. Usually this measurement matrix $\boldsymbol{\Phi}$ is a random one (see Sect. 3). In order to avoid transmitting this random matrix between the sensor nodes and the BS, the authors in [26] proposed a simple strategy. In which the BS constructs the matrix $\boldsymbol{\Phi}$ by using a random seed. Then the BS broadcasts this seed to the sensor nodes, use it to generate the same matrix $\boldsymbol{\Phi}$. The idea behind CS encryption method relies on the fact that the attacker does not have the pseudo-random key (seed) which used to generate the sensing matrix $\boldsymbol{\Phi}$.

But the problems here are remain in, how the BS and the sensor nodes exchange this seed to generate $\boldsymbol{\Phi}$ matrix in secure way? In addition to, CS encryption method is robustly susceptible to CPA attacks [17] . To address these problems, this research paper introduces lightweight CS security scheme (LSS) which has two main parts. First one is key exchange scheme, which allows both of the BS and the sensor nodes to exchange this seed in simple and secure way. Second one is attacks mitigation method, which uses simple way to protect CS method from CPA attacks. For simplicity, we can considered the scenario between one sensor nodes, attacker and the BS (see Fig. 3). The proposed LSS has several advantages:

- There is no pre-store keys process in the sensors.
- The proposed scheme shifted all complex mathematic computation to the BS side.
- All operation used by the sensors are very simple, so no extra energy and storage are needed.

The LSS consists of three stages which are: key generation stage, key exchange stage and data compression with encryption stage. Figure 2 shows the flow chart of LSS.

4.1 Key Generation Stage

In this stage, the BS and the sensor nodes doing the following individually.

At the BS side As shown in Sect. 3, the random matrices like Gaussian or Bernoulli distribution matrix is the best choices for CS method. Any pseudorandom number generator algorithm used an number or vector to initialize its process this number is called seed. This seed can be either random or not. So, if the BS and the sensor node have the same seed they will have the same random sensing matrix $\boldsymbol{\Phi}$ to decrypt and encrypt the data respectively.

But unfortunately if the attacker success to guess this seed it can easily generates the same sensing matrix. So, the LSS takes in consideration the design of this seed to make the guessing process very difficult. Since, there is no power restrictions in BS side, therefor LSS uses two dimensional (2-D) chaotic maps [18] to generate the seed g_S. Chaos: It's a movement which appears in the determined and nonlinear dynamic system also, it seems like random and unpredictable.

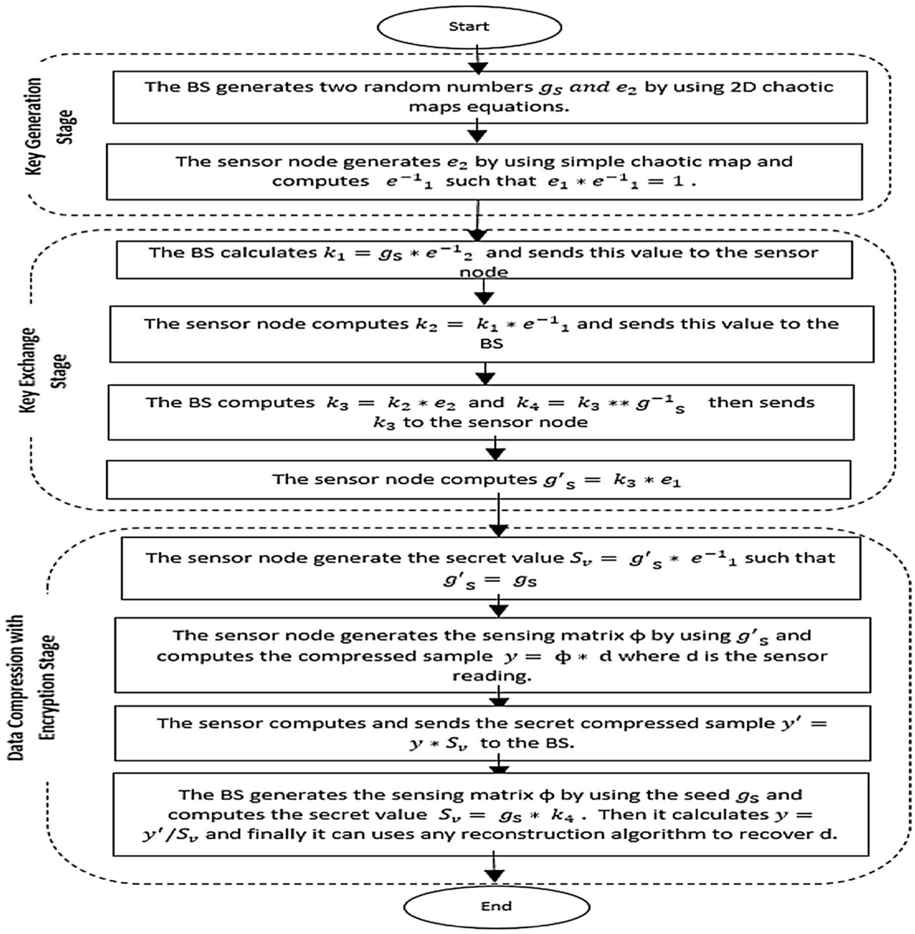
Start
Key Generation Stage
The BS generates two random numbers g_S and e_2 by using 2D chaotic maps equations.
The sensor node generates e_2 by using simple chaotic map and computes $e^{-1}{}_1$ such that $e_1 * e^{-1}{}_1 = 1$.
Key Exchange Stage
The BS calculates $k_1 = g_S * e^{-1}{}_2$ and sends this value to the sensor node
The sensor node computes $k_2 = k_1 * e^{-1}{}_1$ and sends this value to the BS
The BS computes $k_3 = k_2 * e_2$ and $k_4 = k_3 ** g^{-1}{}_S$ then sends k_3 to the sensor node
The sensor node computes $g'_S = k_3 * e_1$
Data Compression with Encryption Stage
The sensor node generate the secret value $S_v = g'_S * e^{-1}{}_1$ such that $g'_S = g_S$
The sensor node generates the sensing matrix ϕ by using g'_S and computes the compressed sample $y = \phi *$ d where d is the sensor reading.
The sensor computes and sends the secret compressed sample $y' = y * S_v$ to the BS.
The BS generates the sensing matrix ϕ by using the seed g_S and computes the secret value $S_v = g_S * k_4$. Then it calculates $y = y'/S_v$ and finally it can uses any reconstruction algorithm to recover d.
End

Fig. 2 Flow chart for operations in LSS

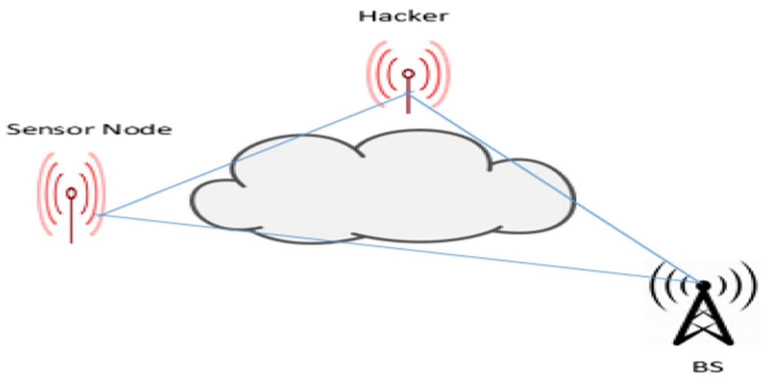

Fig. 3 IoT security model

The sensitivity of its initial conditions can produce a large number of chaotic sequences which are unrelated, similar to the noise and can be regenerated. Two-dimensional logistic dynamic equations are as following [18]

$$x_{i+1} = \mu_1 x_i(1 - x_i) + \gamma_1 y_i^2 \tag{5}$$

$$y_{i+1} = \mu_2 y_i(1 - y_i) + \gamma_2 \left(x_i^2 + x_i y_i\right), \tag{6}$$

where i is the number of iterations to produces two numbers x_{i+1} and y_{i+1}. The values of x_i and y_i are used as initial values to Eqs. (5) and (6) respectively. The dynamical behavior of two-dimensional logistic mapping are controlled by the power equation constant parameters μ_1, μ_2, γ_1 and γ_2. Equations (5) and (6) respectively, increase the quadratic coupling of the items x_i^2, y_i^2 and $x_1 y_i$ and provides more security to the system. When $2.75 < \mu_1 < 3.4$, $2.7 < \mu_2 < 3.45$, $0.15 < \gamma_1 < 0.21$, and $0.13 < \gamma_2 < 0.15$, Eqs. (5) and (6) respectively come into chaotic state which generates a chaotic sequence in the region (0, 1] [18].

At The sensor node side CPA attack can threatens CS method in which the attacker can obtain the ciphertexts y for arbitrary plaintexts x. To protect CS method from this kind of attacks, LSS creates a secret compressed sample y' by multiplying the original compressed sample y with secrets values S_v. The role of this secret value S_v will describe in details in compression and encryption stage. To create S_v the sensor node generates random value e_1 and multiplying e_1^{-1}' with the seed g_s which received from the BS i.e $S_v = g_S \times e_1^{-1}$. In order to generate e_1, the sensor node also uses simple logistic chaotic map equation [17], using a quadratic recurrence equation .

$$c_{n+1} = bd \times c_n \times (1 - c + n) \tag{7}$$

where $bd \in R^{+}_{\neq 0}$, positive real numbers, is called the biotic potential and each value in Eq. (5) depends on the pervious values.

The processes of this stage can be described as following:

1. At node side: the node uses Eq. (7) to generate e_1 such that $e_1 = c_{n+1}$ and then compute its inverse e_1^{-1} such that $e_1 * e_1^{-1} = 1$.
2. At the BS side: the BS uses Eqs. (5) and (6) to generate e_2 and g_S such that $e_2 = x_{i+1}$ and $g_S = y_{i+1}$.

4.2 Key Exchange Stage

The CS encryption process assumes that, only the BS and the sensor node have the same seed to generate the same sensing matrix $\boldsymbol{\Phi}$ for encryption and decryption. But this assumption facing great problem, how can the BS and the sensor node exchange this seed while the hacker tracking the communication channel between them as shown in Fig. 3. To solve this problem, LSS proposes a simple algorithm which called Key Exchange Algorithm as shown in Algorithm 1.

This algorithm allows to exchange the seed between the sensor node and the BS which is safe and simple way. Firstly, the BS calculates $k_1 = g_S * e_2^{-1}$ and sends this value to the sensor node. If the hacker receive this value and use it as a seed, it will give to the hacker a wrong data. Also it will be very difficult process to guess the g_S because LSS uses 2-D chaotic map to generate this number. After that, the sensor node calculates $k_2 = k_1 * e_1^{-1}$ and sends this value to the BS. As k_1, the hacker will face the same difficulty if it use k_2. Then, the BS calculates $k_3 = k_2 * e_2$ and $k_4 = k_3 * g_S^{-1} = e_1^{-1}$ then sends k_3 value to the sensor node which will get the seed when it calculates $g'_S = k_3 * e_1 = g_S$.

Finally, now the BS and the sensor node have the same values of seed g_s and e_1^{-1} and then they can begin the next stage. The procedures of this stage can be shown in Fig. 4.

The mathematical proof for key Exchange Algorithm can be expressed as following:

- As shown in key generation stage the sensor node generates random number e_1 and the BS generates e_2 and g_S. where g_S is the seed which the BS wants to send to the sensor node in safe mode.
- The base station calculates $k_1 = e_2^{-1} * g_S$ and then sends this value to the senor node
- The sensor node uses this value to calculate $k_2 = k_1 * e_1^{-1} = e_2^{-1} * g_S * e_1^{-1}$ and then sends this value to the BS
- The BS calculates $k_3 = k_2 * e_2 = e_2^{-1} * g_S * e_1^{-1} * e_2 = g_S * e_1^{-1}$ and $k_4 = k_3 * g_S^{-1} = e_2^{-1} * g_S * e_1^{-1} * e_2 * g_S^{-1} = e_1^{-1}$ then sends k_3 to the sensor node
- Finally the sensor node calculates $g'_S = k_3 * e_1 = g_S * e_1^{-1} * e_1 = g_S$
- Now, the BS and the sensor node have the same seed $g'_S = g_S$ and same value $k_4 = e_1^{-1}$ and they can start the next stage.

Algorithm 1 Key Exchange Algorithm :

```
1: The BS calculates $k_1 = e_2^{-1} * g_S$
2: The BS sends $k_1$ to the sensor node
3: The sensor node calculates $k_2 = k_1 * e_1^{-1}$
4: The sensor node sends $K_2$ to the BS
5: The BS calculates $k_3 = k_2 * e_2$ and $k_4 = k_3 * g_S^{-1}$
6: The BS sends $k_3$ to the sensor node
7: The sensor node calculates $g'_S = k_3 * e_1$
8: Finally both of the BS and the sensor node have the same values of $g'_S = g_S$ and $k_4 = e_1^{-1}$, where $g_S$ will use as a seed to
   generate sensing matrix $\Phi$
```

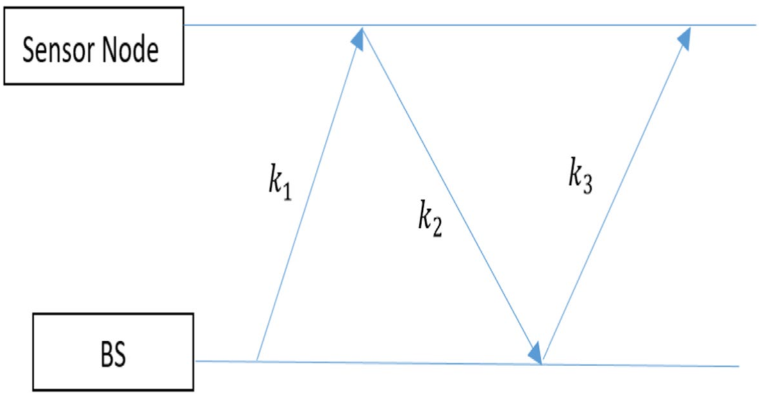

Fig. 4 Key exchange stage procedures

4.3 Data Compression with Encryption Stage

Now, the BS and the sensor node have the same seed g_S such that $g'_S = g_S$ and the same value of e_1^{-1} such that $k_4 = e_1^{-1}$. The sensor node and the BS use Algorism 2 to encrypt and decrypt the sensor data as following: The sensor node uses this seed in order to generate the sensing matrix $\boldsymbol{\Phi}$ to encrypt and compress its data to y by using the Eq. (2). Before the sensor node sends this compressed sample y to the BS, it computes the secret compressed sample y' by multiplying y with a secret value S_v as shown blow [Eq. (8)]:

$$y' = y \times S_v, \tag{8}$$

where $S_v = g'_S \times e_1^{-1}$. Then the sensor node sends y, to the BS.

Algorithm 2 Data Compression with Encryption Algorithm :

```
1: The sensor node computes the secret value $S_v$ such that $S_v = g_S^{'} * e_1^{-1}$
2: The sensor node uses the seed value $g_S^{'}$ to generate the sensing matrix $\Phi$ and then computes the compressed sample $y$
3: The sensor node calculates the secret compressed sample $y^{'}$ such that $y^{'} = y * S$ and sends it to the BS
4: The BS computes the secret value $S_v = g_S * k_4$
5: The BS generates the sensing matrix $\Phi$ by using $g_S$
6: The BS recomputes $y$ such that $y^{'} = y/S$
7: Finally the BS can uses any reconstruction algorithm to recover the sensor data
```

Finally, the BS computes the secret value S_v such that $S_v = g_S * e_1^{-1}$ and recompute the compressed sample y from the secret y' received from the sensor node by using Eq. (9)

$$y = y' / S_v. \tag{9}$$

Then it uses the same seed g_S to generate the same matrix $\boldsymbol{\Phi}$ to encrypt and recover the original data from y by using any reconstruction algorithm like OMP algorithm [23]. Without knowing this seed g_S and the secret value S_v it will be very difficult to generate the same sensing matrix $\boldsymbol{\Phi}$ and so no one can reconstruct y except the BS. To simplify the proposed LSS, an simple example scenario is provided in the section.

5 Example Scenario

This section aims to explain the proposed LSS by providing a simple example. LSS uses the same assumption in Fig. 3, where it have one sensor node wants to send its' data $d = [100010001001]^T$, where d size is 12×1, to the BS and there is an attacker tracking the communication channel between BS and sensor node. The proposed LSS goes as follows:-

Key generation Stage During this stage the senor node uses Eq. (7) to create the number $e_1 = 0.5636363626222726$ such that $e_1^{-1} = 1.774193551579214$ and the BS uses the 2-D chaotic map Eqs. (5) and (6) to generate two random numbers $e_2 = 0.635801265819775$ and $g_S = 0.589524844461205$, where g_S is considered the seed.

Key Exchange Stage In this stage the BS and the sensor node follow Algorithm 1 in which:

- The BS calculate $k_1 = e_2^{-1} * g_S = (0.635801265819775)^{-1} * 0.589524844461205 = 0.927215587878858$ and then sends this value to sensor node.
- The sensor node calculates $k_2 = k_1 * e_1^{-1} = 0.927215587878858 * (0.5636363626222726)^{-1} = 1.645059916938401$ and then sends this value to the Bs

- Then the Bs calculates $k_3 = k_2 * e_2 = 1.645059916938401 * 0.635801265819775 = 1.045931177538809$ and calculate $k_4 = k_3 * g_S^{-1} = 1.645059916938401 * 0.635801265819775 * 1.696281351660332 = 1.774193551579214$ which equal to e_1^{-1} and then sends k_3 value to the sensor node
- Finally, the sensor node computes the seed by solving $g'_S = k_3 * e_1 = 1.045931177538809 * 0.563636362622726 = 0.589524844461205$
- Now $g_S = g'_S$ and $e_1^{-1} = k_4$ so the both side have the same values.

Compression and Encryption Stage The sensor node uses $g'_S = 0.589524844461205$ to generate the sensing matrix the measurement $\boldsymbol{\Phi}_{M\times N}$, where $M = 3$ and $N = 12$ is equal to:

$$\boldsymbol{\Phi}_{M\times N} = \begin{pmatrix} 0.53 & 0.86 & -0.43 & 2.76 & 0.72 & -0.20 & 1.409 & -1.20 & 0.48 & -0.303 & 0.88 & -0.809 \\ 1.83 & 0.318 & 0.34 & -1.34 & -0.06 & -0.124 & 1.41 & 0.717 & 1.03 & 0.293 & -1.147 & -2.94 \\ -2.25 & -1.30 & 3.57 & 3.034 & 0.714 & 1.48 & 0.671 & 1.63 & 0.72 & -0.787 & -1.068 & 1.438 \end{pmatrix}$$

After that, the sensor node creates the secret value $S_v = g_S * e_1^{-1} = 0.589524845 * 1.774193551 = 1.04593117753$. The sensor node computes its measurement y by using Eq. (2), $y = \boldsymbol{\Phi} * d = [0.9424664; -0.1387610; 0.6211614]$ and then computes the secret compressed sample $y' = y * S_v = [1.045931177; -0.1451344672; 0.649692146]$ then sends y' to the BS.

At the BS side, it also generates the secret value $S_v = g_S * k_4 = 1.04593117753$ to obtain the original compressed sample y by using Eq. (9) where $y = y'/S_v = [0.94246644; -0.138761; 0.621161]$. Finally, the BS uses the seed g_S to generate the sensing matrix $\boldsymbol{\Phi}$, since $g'_S = g_S$ then $\boldsymbol{\Phi}$ will be same like sensor node matrix. So by using any reconstruction algorithm the BS can successfully recompute d.

6 Simulation Results

This section provides a simulation results to evaluate the performance of the proposed work. We apply the proposed algorithm to reconstruct the signals collected from 54 sensors located in Intel Berkeley Research Lab [32]. Reconstruction accuracy are given in terms of the Average Normalized Mean Squared Error (ANMSE), which is defined as the average ratio of the $||l||_2$ norm of the reconstruction error to$||x||_2$ over the 500 test samples. For each test sample, we employed an individual observation matrix $\boldsymbol{\Phi}$ whose entries were drawn from the Gaussian distribution with mean 0 and standard deviation 1/N. In the BS side, we use OMP algorithm [23] to reconstruct the plaintexts. The (2-D) chaotic maps parameters $\mu_1 = 3.33$, $\mu_2 = 3.44$, $\gamma_1 = 0.17$ and $\gamma_2 = 0.14$ are used. Finally, we vary the compression ratio from 50 to 80% with increase by 10%.

6.1 Key Space Analysis

The 128-bit key occupied key space up to 2^{128}, which could provide a sufficient security against the brute force attack. Through the proposed LSS, in order to generate the key (seed g_S), the BS station uses the (2-D) chaotic map with initial conditions and parameters $\mu_1, \gamma_1, \mu_2, \gamma_2, x_i$ and y_i. Such that, their precision equal to 10^{-14}, so the key space size reached to 10^{84} which approximately equal to $\cong 2^{280}$, it is bigger than 2^{128}. So the key space for the proposed LSS is large enough to resist the brute-force attacks.

6.2 Data Reconstruction

In this part we apply LSS to reconstruct the signals collected from 54 sensors located in Intel Berkeley Research Lab [32]. In Fig. 5 we give as an example for the distribution of the relative recovery error for the 6*th* sensor node. It is clear that, the reconstruction error is equal to 0% when the sensor node and the BS use g_S as a seed and equal to 30%, 39%, and 31% when using k_1, k_2 and k_3 respectively as a seed. So, by using the proposed key exchange algorithm only the sensor node and the BS can recover the data successfully.

6.3 Key Sensitivity Analysis

A good encryption algorithm should be sensitive to the encryption keys in process of both encryption and decryption. During the encryption process, tiny change of keys receive different cipher data and when decrypt the data, if users use wrong key they receive different data values.

Figure 6a, b shows the decryption process of senor data while making tiny change in the seed value ($g_S = g_S - 0.00000000000001$) and using the correct seed g_S respectively. So, it can be concluded that the algorithm is sensitive to the key, a small change of the key will generate a completely different decryption result and cannot get the correct plain-data.

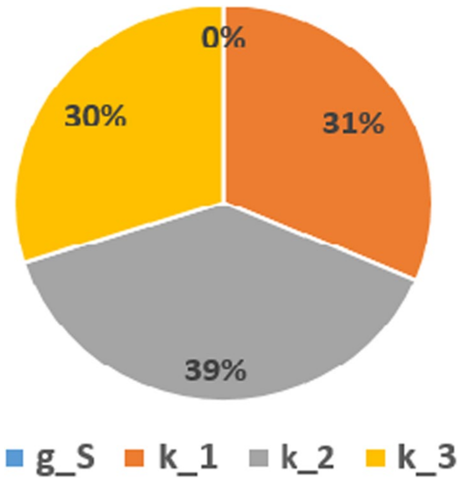

Fig. 5 Reconstruction error result over intel temperature trace

6.4 Network Life Time

In this part we assume that, the network region size is 100 m × 100 m, and the number of sensor nodes ranges from 50 to 200 nodes in the increments of 50 nodes and the base station is located at (x = 50, y = 50).

LSS uses the same energy parameters that used in [27] where to transmit an l-bit message over a distance d, the radio power consumption will be,

$$E_{Tx}(l,d) = \begin{cases} lE_{elec} + l\epsilon_{fs}d^2 & d < d_0 \\ lE_{elec} + l\epsilon_{mp}d^4 & d \geq d_0 \end{cases} \tag{10}$$

and to receive this message, the radio expends will be

$$E_{Rx}(l) = l\, E_{elec} \tag{11}$$

Simulated model parameters are set as: $E_{elec} = 50$ nJ/bit, $\epsilon_{fs} = 10\,\text{pJ/bit/m}^2$, $\epsilon_{mp} = \frac{13}{10000}$ pJ/bit/m^4, $d_0 = \sqrt{\epsilon_{fs}/\epsilon_{mp}}$, and the initial energy per node = $2J$.

We compare the proposed scheme performance with ECC-224 algorithm [4] and RSA-1024 algorithm [3] integrated with CS method . The network life time is measured according to the death of the First node. Figure 7, shows the lifetime and the average energy

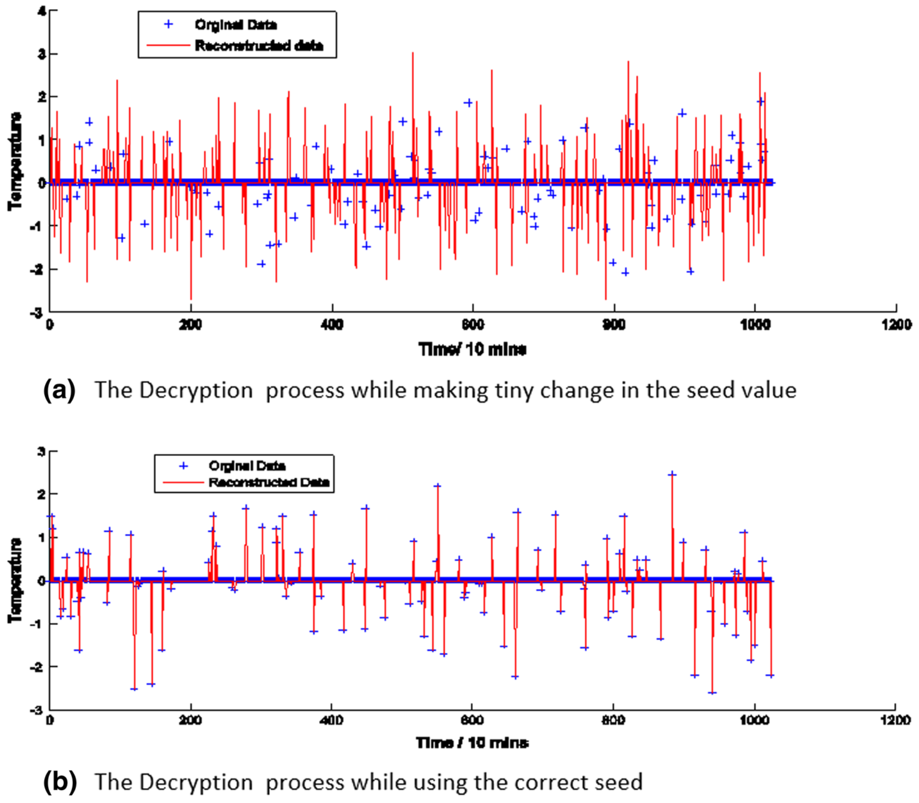

Fig. 6 The decryption process while varying the seed value

consumption per round for our protocol. The same Figure illustrates the effectiveness of the proposed scheme in prolonging network lifetime than its counterparts ECC and RSA algorithms, that is because:

- No extra bits have been added to the sensors' data
- The encryption and compression are processed in the same time
- All complex computations are shifted to the BS side

6.5 CPA Attacks Test

In this section we test our scheme against CPA. We followed the same scenario in [17] in which, it define two types of attackers, first one is oblivious attacker and second one is non-oblivious attacker.

6.5.1 The Oblivious Attacker

In this kind of attackers, we assumed that the attacker can successfully start CPA and it is able to obtain the ciphertext y, which sent by the sensor nodes. But the attacker do not know about the secret value S_v. So, the attacker starts to decrypt y, by using any reconstruction algorithm . As shown in Fig. 8 its reconstruction error result is very high so the attacker unable to decrypt the sensor data.

6.5.2 The Non-oblivious Attacker

This type of attacker can start CPA attacks in addition to the attacker knows about the secret value S_v. LSS assumes that the attacker tries to guess S'_v such that $S'_v = (1 - r) * S$, where $r = [5\%, 10\%, 15\%, 20\%, 25\%]$ is the deviation percentage from the real value of S_v. As shown in Fig. 9 the reconstruction error is still big whatever the value of r. Only when $r = 5\%$ the attacker reach to the minimum reconstruction error but the attacker still unable to decrypted the correct data.

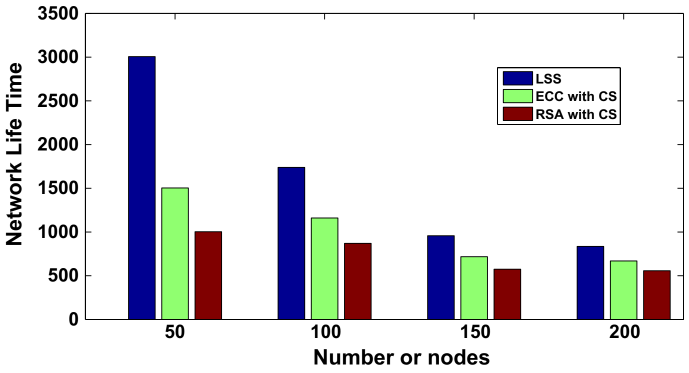

Fig. 7 Network life time as a function of number of sensor nodes

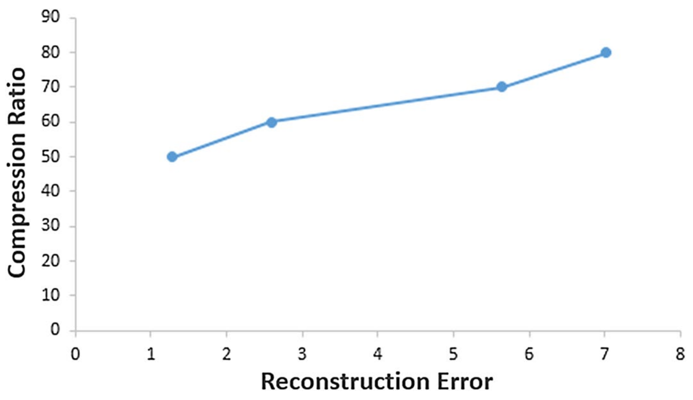

Fig. 8 Compression ratio versus reconstruction error for the the oblivious attacker

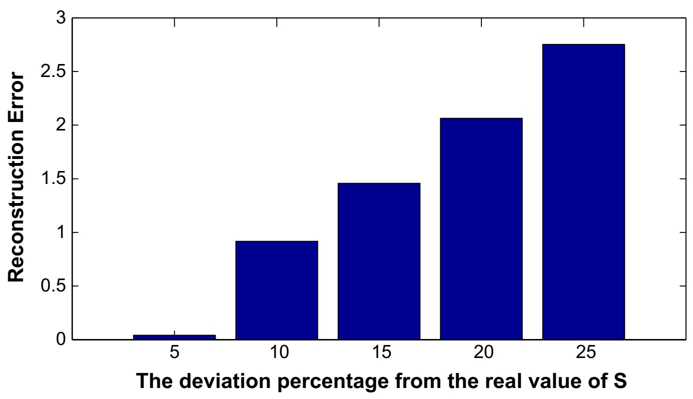

Fig. 9 The reconstruction error for the non-oblivious attacker with in correct values of S_v

7 Conclusion

In this paper, we proposed lightweight secure scheme (LSS) for IoT. The main objectives for the proposed LSS are to secure the IoT and minimize the energy consumption. LSS consists of three stages. Firstly, key generation stage in which the sensor nodes and the BS used two different chaotic maps to generate random numbers and the seed which considered the key. The second one is called Key exchange stage which allows the sensor nodes and the BS to exchange the seed in very simple and secure way. Finally, compression with encryption stage successes to make our scheme immune to CPA attacks by generating secret compressed samples. The results show that, a legitimate users can decrypt data without errors, since they have the correct seed. On the other hand, an attacker decrypts the same data with a significant higher error. Also, it is clear that our scheme success to prolong the network life time comparing with the others encryption algorithms.

References

1. Chen, L., Thombre, S., Jarvinen, K., Lohan, E. S., Alen-Savikko, A. K., Leppakoski, H., et al. (2017). Robustness, security and privacy in location-based services for future IoT: A survey. *IEEE Access*, *5*(99), 1.
2. Palopoli, L., Passerone, R., & Rizano, T. (2011). Scalable offline optimization of industrial wireless sensor networks. *IEEE Transactions on Industrial Informatics*, *7*(2), 328–329.
3. Mollin, R. A. (2006). *An introduction to cryptography*. Boca Raton: CRC Press.
4. Vanstone, S. A., Menezes, A. J., & Oorschot, P. C. (1999). *Handbook of applied cryptography*. Boca Raton: CRC Press.
5. Fragkiadakis, A., Tragos, E., & Traganitis, A. (2014). Lightweight and secure encryption using channel measurements. In *4th international conference on wireless communications, vehicular technology, information theory and aerospace, Aalborg* (pp. 1–5).
6. Donoho, D. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, *52*(4), 1289–1306.
7. Candes, E. J., & Tao, T. (2006). Near-optimal signal recovery from random projections: Universal encoding strategies. *IEEE Transactions on Information Theory*, *52*(12), 5406–5425.
8. Cossalter, M., Valenzise, G., Tagliasacchi, M., & Tubaro, S. (2010). Joint compressive video coding and analysis. *IEEE Transactions on Multimedia*, *12*(3), 168183.
9. Premnath, S., Jana, S., Croft, J., Gowda, P., Clark, M., Kasera, S., et al. (2013). Secret key extraction from wireless signal strength in real environments. *IEEE Transactions on Mobile Computing*, *12*(5), 917–930.
10. Li, Z., Xu, W., Miller, R., & Trappe, W. (2006). Securing wireless systems via lower layer enforcements, In *Proceedings of the WiSe* (pp. 33–42).
11. Dautov, R., & Tsouri, G. (2013). Establishing secure measurement matrix for compressed sensing using wireless physical layer security. In *Proceedings of ICNC* (pp. 354–358).
12. Fragkiadakis, L., Tragos, E., Makrogiannakis, A., Papadakis, S., Charalampidis, P., & Surligas, M. (2016). *Signal processing techniques for energy efficiency, security, and reliability in the IoT domain* (pp. 19–447). New York: Springer.
13. Rachlin, Y., & Baron, D. (2008). The secrecy of compressed sensing measurements. In *Proceedings of 46th annual Allerton conference on communication, control, and computing* (pp. 813–817).
14. Cambareri, V., Mangia, M., Pareschi, F., Rovatti, R., & Setti, G. (2015). Low-complexity multiclass encryption by compressed sensing. *IEEE Transactions on Signal Processing*, *63*, 21832195.
15. Yu, L., Barbot, J. P., Zheng, G., & Sun, H. (2010). Compressive sensing with chaotic sequence. *IEEE Signal Processing Letters*, *17*(8), 731734.
16. Zhang, L., Wong, K., Li, C., & Zhang, Y. (2014). *Towards secure compressive sampling scheme*. CoRR.
17. Fragkiadakis, A., Kovacevic, L., & Tragos, E. (2016). Enhancing compressive sensing encryption in constrained devices using chaotic sequences. In *Proceedings of the 2nd workshop on experiences in the design and implementation of smart objects* (pp. 17–22).
18. Liu, H., Zhu, Z., Jiang, H., & Wang, B. (2008). A novel image encryption algorithm based on improved 3D Chaotic Cat Map. In *The 9th international conference for young computer scientists*.
19. Candes, E., & Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, *52*(52), 145–156.
20. Baraniuk, R. G. (2007). Compressive sensing. *IEEE Signal Processing Magazine*, *24*(4), 118–121.
21. Mallat, S. (1999). *A wavelet tour of signal processing*. Cambridge: Academic Press.
22. Venkataramani, R., & Bresler, Y. (1998). Sub-nyquist sampling of multiband signals: Perfect reconstruction and bounds on aliasing error. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 12–15).
23. Tropp, J., & Gilber, A. (2007). Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, *53*(14), 4655–4666.
24. Donoho, D., Yaakov, T., Drori, I., & Jean, S. (2012). Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Transactions on Information Theory*, *58*(2), 1094–1121.
25. Deanna, N., & Roman, V. (2009). Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of Computational Mathematics*, *9*(3), 317–334.
26. Luo, C., Wu, F., Sun, J., & Chen, C. W.(2009). Compressive data gathering for large-scale wireless sensor networks. In *Proceedings of the 15th annual international conference on mobile computing and networking, MobiCom 09* (pp. 145–156), New York, NY, USA.

27. Heinzelman, W., Chandrakasan, A., & Balakrishnan, H. (2000). Energy efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Annual Hawaii International Conference* (pp. 3005–3014).

Ahmed Aziz received his B.Sc. degree with honor in Computer science in June 2007, the M.S. degree in OCT 2014, in the area of computer science from the Faculty of Computers and Informatics, Benha University, Benha, Egypt. He is doing his Ph.D. in the school of computer and system science, Jawaharlal Nehru University, New Delhi, India. From December 2007 till 2014 he has been working as a demonstrator and from 2014 till now he has been working as Assistant Lecturer at the Department of Computer Science, College of computers and informatics, Benha University, Benha, Egypt (www.fci.bu.edu.eg). His research interests include sensor networks, Compressive sensing, computing, wireless networks, and IoT.

Karan Singh received the Engineering degree (Computer Science and Engineering) from Kamala Nehru Institute of Technology, Sultanpur, UP, India. He is the M.Tech. (Computer Science and Engineering) and Ph.D. (Computer Science and Engineering) from Motilal Nehru National Institute of Technology UP, India. He worked at Gautam Buddha University, UP, India. Currently, he is working with School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi. His primary research interests are in Computer network, Network security, Multicast communication, IoT, and Software define the network. He is supervisor of many researcher scholars. He is reviewer of Springer, Taylor & Francis, Elsevier Journals and IEEE Transactions. He is an Editorial Board Member of Journal of Communications and Network (CN), USA. He published many research papers in refereed journals and good conferences. He organized the workshops, conference Sessions and trainings. Dr. Singh worked as General Chair of the international conference (Qshine 2013) at Gautam Buddha University, India. Recently he organized a workshop on "PYTHON", short term Course (STC) at JNU and special session in ICGCET 2016 at Denmark. He nominated for Who's who in World in the year 2008. Dr. Singh has been joined as a Professional member of ACM, New York, CSTA USA, CSI, Secunderabad, India, CRSI, Kolkata, IEEE, USA, IACSIT, Singapore, ICST, IAENG, Hong Kong, ACEEE, India, ISOC, USA and AIRCC.