Combining Merkle Hash Tree and Chaotic Cryptography for Secure Data Fusion in IoT

Nashreen Nesa(✉) and Indrajit Banerjee

Department of Information Technology, Indian Institute of Engineering Science and Technology, Shibpur, Howrah 711103, West Bengal, India
{nashreennesa.rs2016,ibanerjee}@it.iiests.ac.in
http://www.iiests.ac.in

Abstract. With the wide applicability of sensors in our daily lives, security has become one of the primary concerns in an Internet of Things (IoT) environment. Particularly, user's privacy and unauthorized access to sensitive information needs to be kept in mind while designing security algorithms. This paper puts forward a security protocol that integrates authentication of the deployed IoT devices and encryption of the generated data. We have modified the well-known Merkle Hash Tree to adapt to an IoT environment for authenticating the devices and utilized the concepts of Chaos theory for developing the encryption algorithm. The use of chaos in cryptography are known to satisfy the basic requirements of the cryptosystem such as, high sensitivity, high computational speed and high security. In addition, we have proposed a chaotic map named Quadratic Sinusoidal Map which exhibits better array of chaotic regime when compared to the traditional quadratic map. The security analysis demonstrate that the proposed protocol is simple having low computational requirements, has strong security capabilities and highly resilient to security attacks.

Keywords: Chaos theory · Merkle Hash Tree · IoT · Security · Encryption

1 Introduction

Past researches in the field of Internet security are limited to traditional internet, but with the recent advancement of IoT technologies, these solutions need to be refined so as to cater to the specific needs of IoT [1,2]. Security algorithms for IoT applications should be such that it ensures source authentication, confidentiality, data integrity and resistance against attacks [3]. There is no denying the fact that every smart object in an IoT environment carries the potential of becoming the entry point of malicious activity. Essentially in IoT, security is of paramount importance since this emerging technology revolution entirely depends on the acceptability of its customers. As a result, fusion of data, that is the basis upon which the critical decisions are taken, becomes more challenging if end-to-end security between a sender and a receiver is not ensured. In IoT applications, nodes are constantly communicating highly sensitive data and due to this, such data are vulnerable to security attacks. Authentication is a prerequisite and the most important requirement for secure communication in IoT applications since the communicating devices are prone to security attacks. It is essential for every communicating device in the network to verify its identity so that no unauthorized device can take part in communication [4]. Data confidentiality and integrity are also equally important since malicious alteration of sensor data may result in life-threatening consequences especially in critical IoT applications such as healthcare [5–7]. Data confidentiality that is achieved through encryption of the sensed data is vital in order to ensure that no unwarranted disclosure of sensitive information is possible [8]. Thus, in our work, we have proposed a data fusion approach that ensures the security of the devices as well as the data generated to form useful, reliable and secured result. In our proposed security scheme, Chaos theory is used for encryption and decryption of the data and Merkle Hash Tree for device authentication. Since IoT devices have in-built radio frequency identification (RFID) tags for device identification, we have found its application in our work where it is used for the purpose of authentication by exploiting its uniqueness to serve our purpose. Moreover, we have proposed a sinusoidal chaotic map that is used for encryption; the initial condition and the control parameters of which are produced from the Merkle Hash Tree that forms the basis upon which the maps are created. Our work uses lightweight computation modules, such as one-way hash functions and bitwise exclusive-or operation, for designing the secure data fusion protocol. Besides being a lightweight computation tool, the use of hash operations also preserves anonymity since the hash values are impossible to regenerate. In secure communication, the receiver should also have the provision to examine whether the message has been altered during transmission. For this purpose, this paper adopts a simple mechanism for integrity checking by padding the number of zeros in the original ciphertext. Although researchers have investigated the concept of designing cryptosystem based on chaotic maps in the past, but to the best of our knowledge, this is the first attempt at combining Merkle hash Tree with a novel chaotic map in order to achieve security. Specifically, our contributions are listed as follows:

- First, we present an authentication scheme based on the Merkle hash tree technique where the hash values of the leaves are calculated on the unique RFID tags attached to IoT devices.
- Second, we propose a novel Quadratic Sinusoidal chaotic map whose dynamical characteristic properties are studied and confirmed to belong to the chaotic community.
- Third, an efficient data fusion protocol that is based on the Merkle Hash Tree and the chaos theory is proposed. The Merkle Hash Tree generates the initial conditions and the control parameters of the chaotic map that are used for

Table 1. Common Notations used in this work

Notation	Description
D_i	ith IoT device
n	Number of devices in the network
TC	Trusted Data Fusion Centre
l	Number of levels in MHT
$\phi_{i,j}$	Merkle hash assignment of the i^{th} node at the j^{th} Level of MHT
H(.)	Secure one-way Hash operation i.e. SHA-1
$\oplus$	XOR operation
$\|\|$	Concatenation operation
l	Number of levels in Merkle Hash Tree
θ	Merkle Hash Path
$\mathcal{S}$	Pre-shared Session key
$\mathcal{K}$	Initial condition of the map; also serves as the key
Itr	Number of iterations in the map
$\mathcal{P}_i^t$	Plaintext from the i^{th} device at the t^{th} instant
$\mathcal{C}_i^t$	Ciphertext from the i^{th} device at the t^{th} instant
$N^{(0)}$	Zero count in ciphertext $\mathcal{C}_i^t$
$Cipher$	Final ciphertext after appending $N^{(0)}$ i.e., $\mathcal{C}_i^t \| N^{(0)}$

encryption/decryption of messages. After which they are effectively fused to derive the intended result.

– Lastly, extensive security analysis indicates that the proposed scheme can resist all kinds of attacks in addition to ensuring data integrity, confidentiality and authenticity.

The remainder of the paper is organized as follows: a related study on the recent trends in research is presented in Sect. 2, followed by the introduction of concepts of Merkle Hash Tree with its key definitions in Sect. 3. Details about our proposed Modified Sinusoidal Quadratic map is presented in Sect. 4. Next, in Sect. 5, a description of all the phases in our proposed architecture is given. Section 6 describes an experimental scenario of our proposed algorithm for easy understanding, followed by the security analysis of the algorithms in Sect. 7. Finally, the paper is concluded with its ending remarks in Sect. 8.

2 Related Works

Developing security solutions that fulfils the specific requirements of IoT environment is a challenging task and currently a lot of researchers are focussing on this domain. Owing to the distributed network of the IoT devices, security solutions are often integrated with cloud servers and could computing technologies

[9–11]. Specifically, in [9], a mutual authentication scheme is presented based on Elliptic Curve Cryptography (ECC) for secure communication between devices and cloud servers. The proposed protocol has been verified using AVISPA tool to be highly efficient with low computational cost. Since, conventional cryptography solutions are not applicable for IoT applications and schemes developed for IoT must be lightweight. Owing to this requirement, the authors in [10] propose a light weight authentication protocol for IoT enabled devices. For mutual authentication, BAN logic has been used and the protocol was simulated using AVISPA software. Another closely related work is presented in [11], where a robust authentication scheme for resource-constraint IoT devices with cloud assistance is designed. The proposed scheme is lightweight since only cryptographic modules such as one-way hash functions and XOR operations are used. Moreover, security in terms anomaly detection for specific applications can be found in the available literature. For instance, in [12], the authors have introduced a secured IoT-based traffic system with intelligence that is capable of analyzing traffic data into good or bad using Support Vector Machine (SVM). The system was implemented using Raspberry Pi3 and Scikit. Similar to this work, the authors in [13] used four machine learning algorithms for detecting forest fire based on real-world dataset. In addition, an IoT architecture is designed that distinguishes cases when the sensors are faulty and when a fire is detected. Subsequent measures and alert through the use of IoT technologies have also been incorporated. Another closely related work is that presented in [14] where the authors proposed a sequence based learning algorithm to detect inconsistent data in IoT devices. The proposed algorithm was tested for both faulty node detection referred to as "Error" and any abnormal activity known as "Event" using three different real-life datasets. Inspired by the human immunity system under pathogenic attacks, a bio-inspired security solution is proposed in [15]. The proposed solution leverage supervised k-mean-based learning to first distinguish the faulty nodes from the good ones, after which it introduces virtual antibodies to deactivate the fraudulent nodes in the system. Owing to the limited computational capabilities of IoT objects and with the intention of helping designers estimate the cost of implementing security solutions, [16] is proposed. Here, the authors presented a formal framework based on the process calculus IoT-LySa [17] to ascertain a trade-off between security solutions and their cost.

Our work is an extension of our previous work [18] that uses chaotic cryptography for developing an efficient light-weight encryption algorithm. Chaos is a popular theory in numerous natural and laboratory systems encompassing several scientific and research areas as a result of which there is a rich body of literature dedicated to chaos theory. A review on recent enhancements of traditional data encryption procedures is given in [19].

Particularly, the work in [20] relates to body area networks (BANs) application where the authors used image encryption for testing under the assumption that a significant part of sensor data are of images. An efficient flood forecasting model is presented in by studying the data from an area in Brazil through a WSN network. The data was modelled using machine learning techniques and chaos theory. Moreover, a lot of applications have adopted chaos theory for either encryption, detection or authentication [21,22] ranging from pipe leakage detection [23], flood forecasting [24,25], Iris recognition [26], network traffic forecasting [27] to name a few.

In this work we have used the popular Merkle Hash tree (MHT) as an authentication algorithm. MHT has been extensively adopted by researchers in both IoT [28–30] and non-IoT [31–34] related fields. Focussing on MHT, the authors in [28] proposed an authentication scheme for securing smart grid communication. The authentication protocol is based on Merkle Hash Tree where the authors demonstrated that the proposed scheme incurs less computation cost compared with RSA-authentication mechanisms. Security analysis presented by the authors shows that it can resist replay attack, message injection attack, message analysis attack, and the message modification attack. However, not much is discussed about providing integrity and confidentiality in the process of communication. Next, in [29] also proposed a Neighborhood Area Network (NAN) for authenticating power usage power data in smart grids. The authors incorporated digital signature schemes for fault tolerance. In addition, fault diagnosis schemes are also deployed to pinpoint the errors and reduce the computational and communication load in the system. Another work involving smart grids is presented in [30] to provide mutual authentication authenticate between smart meters and the utility servers. A key management protocol is also presented and the whole system is capable to resisting numerous cryptographic attacks.

As can be observed from the available literature both MHT and Chaos theory are highly efficient standalone secularity tools, the combination of which has never been attempted before. Therefore, the need for developing a security algorithm that amalgamates the advantages of both the theories, motivated this research work. Even though a lot of research have been done on security in IoT, to the best of our knowledge, this paper is the first attempt at combining Merkle Hash Tree and chaotic cryptography for developing a security protocol for IoT environment. Preliminary definitions and notions of both MHT and Chaos theory are briefly discussed next.

3 Merkle Hash Tree

Merkle Hash Tree (MHT) was first introduced in 1989 by Merkle [35] and has since then been used for verification and integrity checking by various applications. It is a popular technique among the Git and Bitcoin community for authenticating users. MHTs have mostly been used as authentication schemes [28,31]. A typical MHT is a binary tree in which the nodes of the tree are simple hash values. The nodes at the lowest level (leaf nodes) could be an arbitrary hash values or the hash values generated from pseudorandom numbers whereas the nodes at the intermediate levels are the hashes of their immediate children. The root of the MHT is unique since the collision resistance property of hash function ensures that no two hash values differing by atleast 1 bit should be same. To adapt to an IoT environment, the traditional definition of MHT concepts have been modified.

Definition 1 ***Merkle Hash Assignment.*** *Let T be a MHT created in an IoT setup with n devices, i.e., having* $\log_2 n+1$ *levels and let* $RFID_i$ *be the RFID of* i^{th} *IoT device in T. The Merkle hash assignment associated with the device with* $RFID_i$ *of T at level 1, denoted as* $\phi_{i,1}$ *is computed as*

$$\phi_{i,1} = H(RFID_i) \tag{1}$$

Similarly, the hash values of all node i except for the leaf nodes at level j denoted as $\phi_{i,j}$ *is computed by the following function:*

$$\phi_{i,j} = H(\phi_{2i-1,j-1} || \phi_{2i,j-1}) \tag{2}$$

where '||' denotes the concatenation operator and H(.) is the hash function.

According to Definition 1, the Merkle hash value associated with a device D_i is the result of a hash function applied to its RFID tag. To ensure the integrity of the Merkle hash value of the Trusted Centre (TC), we assume that the root value ϕ_{root} is tamper-resistant whose credentials have been thoroughly checked by top-level security system. Each leaf node in the constructed MHT can be verified through its Merkle Hash Path θ which is defined next.

Definition 2 ***Merkle Hash Path.*** *For each level* $l < \log_2 n + 1$ *(height of the tree), we define Merkle Hash Path* θ *to be the* ϕ *values of all the sibling nodes at each level* l *on the path connecting the leaf to the root node. The Merkle Hash Path signifying the authentication data at level* l *is then the set* $\theta_l | 1 \leq l \leq \log_2 n + 1$.

The authentication procedure of a leaf node is then carried out as: The ϕ value at the leaf is first hashed with its sibling θ_1, which, in turn, is hashed together with θ_2, and so on till the root is reached. At this stage, the calculated root value accumulated through the Merkle Hash Path is compared with equal to the known root value $\phi_r oot$. If it turns out to be equal, then the leaf node is accepted as authentic. It is obvious that consecutive leaf nodes share a large portion of the authentication data θ when the leaves are ordered from left to right in the tree, thereby saving a lot of communication overhead in sending redundant data.

4 Modified Sinusoidal Quadratic Map

Chaotic maps are defined as mathematical functions that characterises the chaotic behaviour of the system and which depends on their initial conditions and control parameters. Our proposed chaotic map inspired by the classical quadratic map [36] is given as

$$x_{n+1} = 1 - \sin(r + a x_n^2) \text{ for } a > 3 \tag{3}$$

where the initial condition x_0, a and r are the control parameters. For our proposed equation, the value of a must be above 3 i.e., $a > 3$ in order to be chaotic.

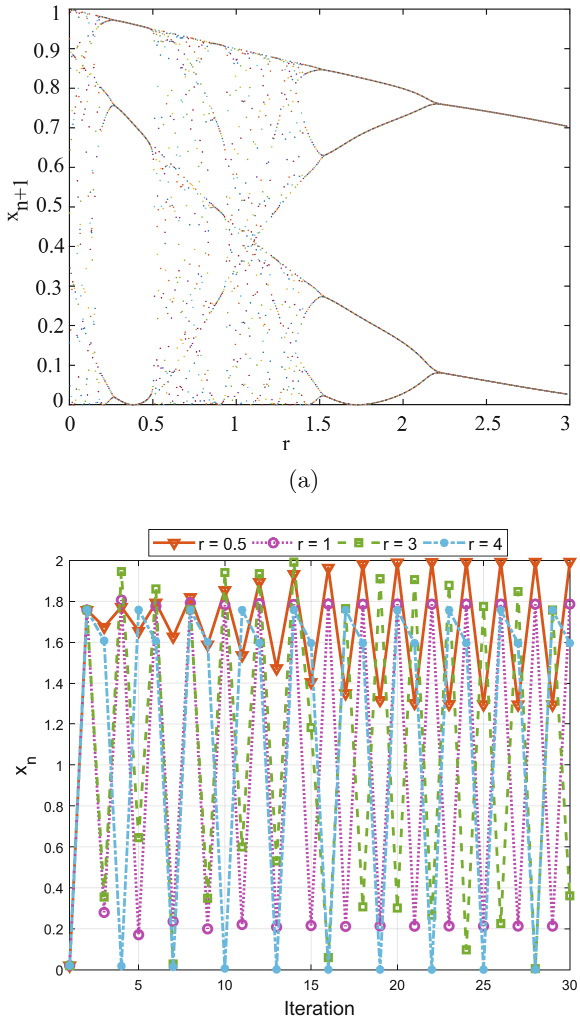
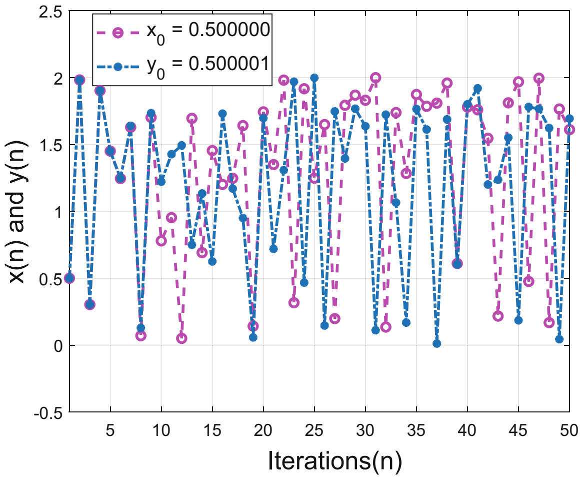


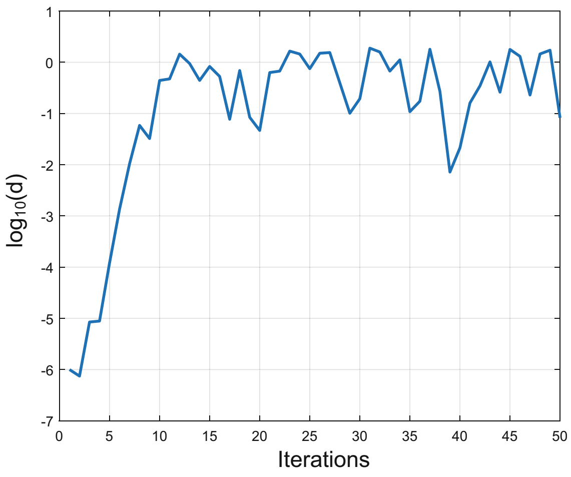
Fig. 1. (a) The proposed sinusoidal chaotic map for a = 4 (b) Convergence and period doubling plot for a = 3 and x_0 = 0.02

4.1 Analysis of the Proposed Map

Bifurcation plot in a dynamical system is a visual representation defining the behaviour of a system when its control parameters undergo some change [37]. The bifurcation diagram of the proposed map is shown in Fig. 1(a) where the solution was iterated for different values of r for a particular value of $a (a = 4)$. From the figure, the three regions of chaos, convergence and bifurcation can be observed clearly. The convergence region can be seen to start at $r = 2.2$, the



(a)



(b)

Fig. 2. (a) Sensitivity to initial conditions for two very close initial conditions of $x_0 = 0.500000$ and $x_0 = 0.500001$ for a = 4 and r = 3.9 (b) Semi-log plot for conditions of $x_0 = 0.500000$ and $x_0 = 0.500001$ for a = 4 and r = 3.9 (Color figure online)

bifurcation region lies in the range $r \in [1.5, 2.2]$ while the region $r \in [0, 1.5]$ can be considered to be chaotic with slight windows of stability that occurs in $r \in [0.3, 0.5]$. Each of these points are plotted for the particular value that the system settles towards for a specific value of r. Figure 1(b), on the other hand shows the phenomenon of period doubling where it can be seen that for $r = 0.5$, the system just starts becoming unstable. This is can correlated with the status

Algorithm 1. Authentication Algorithm using MHT

```
Input: Number of devices in the network n
Output: Authenticated Result

/* Construction of Merkle tree */
1  Number of levels $l \leftarrow \log_2 n + 1$
2  foreach level $j \in l$ do
3      foreach device $i \in N$ do
4          if $\phi_{i,j}$ is a leaf node then
5              Calculate $\phi_{i,j} \leftarrow H(RFID_i)$
6          else
7              $\phi_{i,j} \leftarrow H(\phi_{2i-1,j-1} || \phi_{2i,j-1})$

/* Authenticating an IoT device with RFID X */
8  if $\phi_{i,j} == \phi_{root}$ then
9      X is authenticated to be a valid device
10 else
11     A threat is detected
```

of the status at r = 0.5 in Fig. 1(a) where the diagram shows moderate chaos. Similarly, increasing the values of r results in increase in randomicity until r reaches 3 where the system fluctuates between two values that are the period attractors.

Sensitivity to initial conditions is another prime characteristic of chaos which states that two very close values of initial conditions diverge significantly over time. This phenomenon can be observed in Fig. 2(a) where two very close initial conditions $x_0 = 0.500000$ (in pink) and $y_0 0.500001$ (in blue) are iterated in the chaotic regime. As can be observed from the figure, the two trajectories are almost identical for the first 9 iterations. However, after the 10th iteration, the minuscule difference in the initial conditions diverges exponentially and show little in common as the number of iterations increases. This phenomenon is known as sensitivity to initial conditions and can be observed in our proposed chaotic map thus confirming its ramdomicity. Furthermore, a semi-log plot is constructed and presented in Fig. 2(b) that highlights the difference between the changes in the two initial conditions x_0 and y_0 over time. The difference d_n is calculated as $d_n =| x_n - y_n |$ and its logarithm values and plotted against the number of iterations n in Fig. 2(b). The exponential increase in the value $\log_{10} d$ can be clearly observed from the figure over the passage of time. This plot again gives an indication of the random nature of our map as the number of iterations increases which has been effectively exploited in our work for designing the encryption algorithm.

5 Proposed Security Protocol

Since our proposed architecture relates to an IoT scenario, adapting the known concepts of MHT and chaos theory to an IoT environment was necessary. The processes are explained in detail next.

5.1 Registration

Registration is the first phase where all the n IoT devices $D_i(i = 1, \ldots n)$ wishing to form a network register themselves with the Trusted centre (TC) with their designated n unique RFID tags. TC then constructs the tree by deriving the hash value of each $RFID_i$ and stores them in a table for future authentication. TC then distributes the hash values to all leaf nodes. Thus, only the authenticated devices in the network are in possession of the hash values that is required for their transmission of data.

5.2 Authenticating the Devices Using MHT

As mentioned earlier, authentication is performed by TC which is assumed to be secured from any form of attacks and whose credentials are verified from the top-level security system. Any device D_i wishing to initiate data transfer has to first test for its authenticity. This is done by sending a request message REQ_i to TC that indicates its desire for communication. TC, on receiving REQ_i, asks for the proof that D_i belongs to the network and is a valid device. D_i now sends hash values of the merkle hash path (θ) for authentication. On receiving the proofs, TC calculates the hash value using Eq. 2, and checks if its stored hash of the root (i.e., h_{root}) is equal to the calculated hash. If the two hashes match, the device D_i is an authenticated device and can proceed to the process of data exchange. Algorithm 1 illustrates the process of authentication for our proposed system where each device presented as a leaf node is authenticated by recursively computing and concatenating the hash values along the Merkle Hash path. Since only the hash functions are computed, the computation cost of verification is very low.

5.3 Establishment of Keys

In our work, the key of encryption algorithm is produced by our proposed chaotic map because of its marked nature of randomness. It is a known fact that the more random a key is, the more difficult it is for an attacker to break. Therefore on the basis of the bifurcation diagram, it is easy to note the areas the map produces random chaotic behaviour. The control parameters needed to generate the bifurcation diagram comes from the MHT. Our proposed chaotic map takes as input three control parameters: a pre-shared session key $\mathcal{S}$, a key $\mathcal{K}$ which acts as the initial condition of the map and the number of iterations Itr that signifies the number of times $\mathcal{K}$ is iterated in the map. Based on the number of levels l, keys are produced. These keys are nothing but the value of ϕ at each node in the merkle hash path θ for the device D_i. After which, $\mathcal{K}$ is calculated as follows

$$\mathcal{K} = key_1 \oplus key_2 \oplus key_3 \oplus ... \oplus key_l$$

The value of $\mathcal{K}$ in binary is converted into decimal that serves as the initial condition in the chaotic map. The pre-shared session key $\mathcal{S}$ is generated which is a random number such that the value > 3. This is set keeping in mind that our proposed chaotic map is chaotic in this range. The iteration number Itr is calculated using both the values of $\mathcal{K}$ and $\mathcal{S}$. The number of digits in $\mathcal{K}$, say, dig is estimated. Now, in the generated value of $\mathcal{S}$, dig digits after decimal is extracted and summed up with the value of $\mathcal{K}$ that yields the iteration number.

Theorem 1. *The key space size of our proposed encryption algorithm is* $2^{280} \times l$ *where* l *denotes the number of levels in the Merkle Hash Tree.*

Proof. Since the hash function used in our protocol is SHA-1, which produces a 160-bit binary output, the key space required for $\mathcal{K}$ alone is 2^{160}. Furthermore, in order to ensure the dynamical system falls in the chaotic regime, the range of a that is also the pre-shared session key is restricted to 32-digits values for $a > 3$. This value a or $\mathcal{S}$ is a 32 bit decimal number that is generated randomly. Since the ASCII table supported by MATLAB is composed of 128 values. Each of these 40-hex digits (output of SHA-1) are mapped to its corresponding ASCII, the maximum value of which is 128. Therefore, the maximum value of the hash value at each node, i.e., key_1, key_2,...,key_l (l is the number of levels), cannot exceed $40 \times 128 = 5120$ which requires $\approx$ 13-bits each to represent. Therefore, the value of $\mathcal{K}$ which is the XOR operation of $key_1, key_2, ..., key_l$ also comprises of 13-bits. Hence, the iteration number Itr that is dependent on the number of digits in $\mathcal{K}$ should also not exceed $13 + 1$ (for carry) bits. Summing it all up, the key size for our proposed algorithm is $(2^{160} \times l) \times 10^{32} \times 2^{13} \approx 2^{280} \times l$, where l is the number of levels in the Merkle Hash Tree (Fig. 3).

5.4 Data Encryption/Decryption

The value of ϑ obtained after the iteration process of the chaotic map is then combined with the plain text $\mathcal{P}$ using a XOR operation along with the previous ciphertext value. The inclusion of previous ciphertext for XOR operation was adopted for ensuring dynamic feedback in our proposed architecture. Thus, at any instant t the encrypted data will be given by $Cipher = \mathcal{P}^t \oplus \vartheta \oplus Cipher^{t-1}$. Algorithm 2 displays the essential steps of our chaos based encryption/decryption algorithm. This function takes as an input a 256-bits plaintext $\mathcal{P}^t$ data. In order to add provision for integrity check, an alternative coding approach that appends a count of the '0' bits $N^{(0)}$ in C^t before communicating it to TC. The new message, $\mathcal{C}$, would be only be $\log_2$ 256-bit = 8 bits longer than the original 256-bit message, $Cipher^t$. After appending the zero count $N^{(0)}$ the final ciphertext $\mathcal{C}$ is

Algorithm 2. Proposed Chaotic Encryption algorithm

```
   Input: Raw data $\mathcal{P}$
   Output: Encrypted data $\mathcal{C}$
1  Generate the keys $key_1, key_2, .., key_l$ based on the number of levels $l$
2  Final key $\mathcal{K} \leftarrow key_1 \oplus key_2 \oplus, ...key_l$
3  Convert $\mathcal{K}$ into binary
4  $dig \leftarrow$ number of decimal digits of $\mathcal{K}$
5  $\delta \leftarrow$take $dig$ digits after decimal from pre-shared session key $\mathcal{S}$
6  Iteration $Itr \leftarrow \mathcal{K} + \delta$
   /* Set initial condition $\mathcal{K}$ and $\mathcal{S}$ as control parameter and iterate
      in the chaotic map $Itr$ number of times                                  */
7  ChaosVal $\leftarrow$ Chaos($\mathcal{K}, \mathcal{S}, Itr$)
8  $\vartheta \leftarrow$ ChaosVal $\times Itr$
9  if $\mathcal{P}$ is the first plaintext after registration then $Cipher^t \leftarrow \mathcal{P}^t \oplus \vartheta$
10 else $Cipher^t \leftarrow \mathcal{P}^t \oplus \vartheta \oplus Cipher^{t-1}$
11 $C \leftarrow C^t|\mathcal{N}^{(0)}$
12 return $\mathcal{C}$
```

sent to TC for data fusion through the communication medium. TC, on receiving $\mathcal{C}$, first checks whether the message has been tampered with by comparing the last 8 bits that signifies $N^{(0)}$ with the number of zeros in the first 256-bits in $\mathcal{C}$. If the values do not match, a security threat is detected and subsequent actions are undertaken to remedy the problem. If however, the values match, the ciphertext is assumed to be free of any tampering by an intruder and thus is further processed to extract the plaintext. In our work, since the merkle hash values ϕ is used for modulation in the chaotic map, which is known to both D_i and TC, both can generate the chaotic initial value $\mathcal{K}$ and the Itr value individually. In the decryption process, utilizing the symmetric property of XOR operation TC decrypt the received data $Cipher^t$ as $Cipher^t \oplus \vartheta \oplus Cipher^{t-1} = (\mathcal{P}^t \oplus \vartheta \oplus Cipher^{t-1}) \oplus \vartheta \oplus Cipher^{t-1}$ which equals $\mathcal{P}^t$.

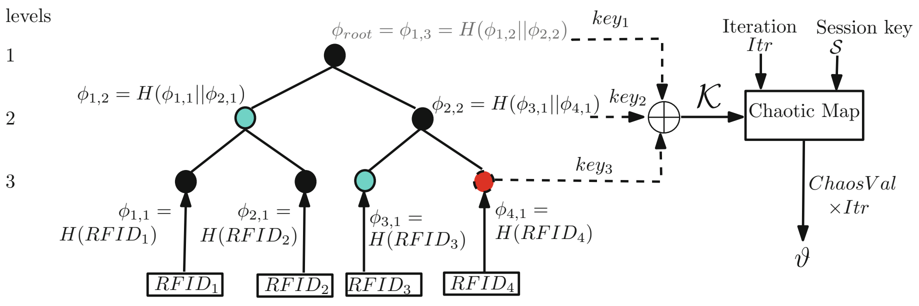


Fig. 3. Key generation using Merkle Hash Tree together with the proposed chaotic map

5.5 Secure Data Fusion

Since the main aim of our work is security in the process of data fusion. Therefore the authentication, key establishment, encryption processes are extended to all the IoT devices in the network that participate in the data fusion process. TC is the centre for data fusion, where encrypted data from all the devices arrive. Based on which the process of fusion takes place to arrive to a conclusion. After which, the decision is communicated to the concerned authority. For instance, in real-time monitoring environment, any critical event needs to be conveyed at real time. The assessment of the critical event is done by fusion of the data appropriately at any time instant at the same time ensuring its security in the process. This idea can be described as follows: assume P_i^t, C_i^t represent t^{th} plaintext and ciphertext respectively from the i^{th} device in the network. Since, a major portion of an device's energy is consumed during the process of communication, reducing the data transfer in the network is useful in saving battery life of energy-constraint IoT devices. To minimize the number of transmissions from thousands of devices towards the TC, a single session key $\mathcal{S}$ is used for all devices for a particular session; after which it becomes obsolete. This limits the number of transmission in the network as well as ensures the security since $\mathcal{S}$ is not the only control parameter that is needed to construct the chaotic map. Thus, TC, on receiving the encrypted message from n devices, decrypts it and performs its computations to derive its result. The data fusion algorithm used by TC is beyond the scope of this paper and thus is avoided to facilitate easy understanding.

6 Experiment

For the sake of simplicity in our experiment, we have simulated an IoT environment consisting of 8 IoT devices $D_i | i = 1, 2, .., 8$ in MATLAB, each generating time varying data P^t every t time instant. For instance, sensory data (referred to as the plaintext) generated by device D_2 is given as $\mathcal{P}_2 = P_2^1, P_2^2, \ldots, P_2^{t-1}, P_2^t$. For our proposed chaotic map $x_{n+1} = 1 - sin(r + ax_n^2)$ for $a \in \{1, 4\}$, the control parameters are the pre-shared session key $\mathcal{S}$, the initial condition x_0 denoted as $\mathcal{K}$ and the iteration number Itr.

Registration

- Step 1: All the 8 IoT devices in the network register themselves with the trusted data fusion centre (TC) with their designated RFID tags, $RFID_i | i = 1, .., 8$. RFIDs are 96-bit binary numbers or 24 hex digits. For our experiment, we have used random 24 hex numbers as RFIDs as shown in Table 2.
- Step 2: TC creates a Merkle Hash Tree with all the devices in the network, in which all the leaf nodes the RFIDs of the devices. The tree is constructed as shown in Fig. 4.

Table 2. RFID tags corresponding to each device for our experiment

Device	RFID tags
D_1	45 3d 6c e1 48 16 85 57 e3 29 c5 89
D_2	77 c0 23 0e b5 0e 39 63 3a 48 5b bf
D_3	2e e0 62 6d 14 ca e6 83 18 7a e7 9e
D_4	ba 9d 08 f4 2b 4b 5e 23 51 d5 70 2a
D_5	5a 3e ed 7e b4 7f d3 e8 60 40 77 37
D_6	3b 0a f1 4a 7c e9 14 ca ac da 3f c9
D_7	21 f5 cb a5 80 0f 82 58 aa 90 f2 d5
D_8	b4 e0 72 d6 50 72 3a cd 67 85 5c ab

– Step 3: In addition, TC maintains a table where RFID tag of each device is stored. A randomly generated pre-shared session key $\mathcal{S} \in \{1, 4\}$ is stored for each time instant t. This session key is used by all the devices for communication for each session, at the expiration of which, the session key $\mathcal{S}$ becomes obsolete.

Authentication using MHT

– Step 4: Device D_8 deciding to initiate a communication does so by sending a request message REQ_8 to TC. Figure 4 depicts this situation where device D_8 is denoted by a red circle.
– Step 5: TC on receiving REQ_8, asks for the proof that D_8 belongs to the network and is a valid device.
– Step 6: D_8 now sends hash values of the merkle hash path for authentication. That is, D_8 sends the value of $\phi_{8,1}$, $\phi_{7,1}$, $\phi_{3,2}$ and $\phi_{1,3}$ as authentication proofs to the TC. The proofs/sibling nodes are highlighted in blue and the initiator node $\phi_{8,1}$ in red in Fig. 4.
– Step 7: TC, on receiving the proofs calculates the resultant hash value from the individual hash values received from D_8 according to Eq. 2, as

$$\begin{aligned}\phi_{1,4} =& H(\phi_{1,3}||\phi_{2,3})\\ =& H(\phi_{1,3}||H(\phi_{3,2}||\phi_{4,2}))\\ =& H(\phi_{1,3}||H(\phi_{3,2}||H(\phi_{7,1}||\phi_{8,1})))\end{aligned}$$

– Step 8: TC now compares the resultant value $\phi_{1,4}$ with its own hash ϕ_{root}. The $\phi_{1,4}$ obtained in the previous step matches with the stored value of ϕ_{root} in our experiment. Since the two hashes are equal, the device D_8 is an authenticated device and thus can proceed to the process of data exchange.

Key Generation and Exchange

– Step 9: Based on the number of levels in the Merkle hash tree, a unique key is generated at each level denoted as $key_1, key_2, ..., key_n$ which is nothing but

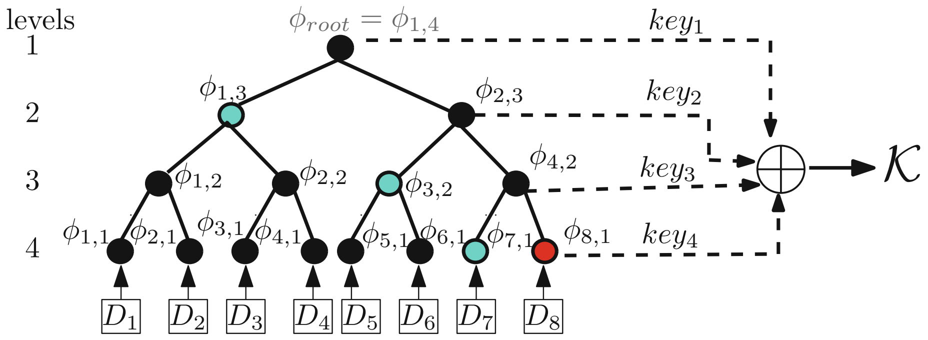


Fig. 4. Key generation using Merkle Hash Tree together with the proposed chaotic map (Color figure online)

the hash values of each node in the Merkle Hash Path. In our experiment number of levels is 4, therefore four keys are generated.

– Step 10: The values of keys i.e., key_1, key_2, key_3 and key_4 are not the proofs that are communicated in the authentication process, but rather the ϕ values of those nodes that fall in the path from the device to the TC. That is, for the device D_8, the keys as illustrated in the the Fig. 4 are $\phi_{8,1}$, $\phi_{4,2}$,$\phi_{2,3}$ and $\phi_{1,4}$ after converting them to their ASCII and summing the resultant values.
– Step 11: All 4 keys key_1, key_2, key_3 and key_4 are XORed to produce the final key $\mathcal{K}$, after which it is converted into decimal that serves as the initial condition for our chaotic map. In our experiment, value of $\mathcal{K}$ (note that the XOR operations are all done in binary, in order to save space, the values are replaced by their decimal equivalent) is

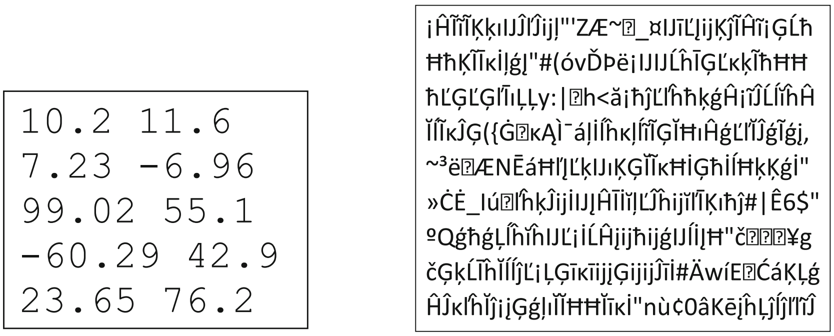
$$\begin{aligned}\mathcal{K} &= 2787 \oplus 2736 \oplus 2671 \oplus 2907\\ &= 2620\end{aligned}$$

The value of $\mathcal{K} = 2620$ is the initial condition for the map generation.
– Step 12: At this point, a shared session key $\mathcal{S}$ is generated between the TC and D_8. $\mathcal{S} \in \{1, 4\}$ is generated such that it falls in the chaotic region of the proposed map. For our experiment we have randomly generated the value as

$$\mathcal{S} = 3.0362054645733205227031703543616$$

– Step 13: The value of Itr that denotes the number of iterations is calculated next. The device/TC first estimates the number of digits in $\mathcal{K}$ as dig. In our experiment, $dig = 4$. Adding dig number of digits after decimal of the value $\mathcal{S}$ to the value of $\mathcal{K}$ yields the value of Itr. That is,

$$\begin{aligned}Itr &= 4 \text{ (value of dig) digits after decimal of } \mathcal{S} + \mathcal{K}\\ &= 0362 + 2620\\ &= 2982\end{aligned}$$



```
10.2 11.6
7.23 -6.96
99.02 55.1
-60.29 42.9
23.65 76.2
```

(a)

```
¡ĤĩĩĨĶķIJĴĪĴijļ"'ZÆ~_¤IJīĽļijĶĵĨĤĩ¡ĢĹħ
ĦħĶĨĨĸİļģļ"#(óvĎÞë¡IJIJĹĥĪĢĽĸķĨħĦĦ
ħĽĢĽĢĺĩıĻĻy:|h<ă¡ħĵĽĺħħķģĤ¡ĩĴĹĺħĤ
ĬĺĨĸĴĢ({ĠĸĄ¯áļİĺħĸļĺĩĨĢĬĦıĤģĽĬĴģĨģį,
~³ëÆNĒáĦľĻĽķIJıĶĢĨĨĸĦİĢħİĺĦķĶģİ"
»ĊĖ_Iú ĺħķĴijİIJĻĤĪĩļĽĴħijĭĺĨĶıħĵ#|Ê6$"
ºQģħģĻĺħĩħIJĽ¡İĹĤjijħijģIJĺİĻĦ"č¥g
čĢķĹĨħĬĺĵĽ¡ĻĢĪĸĩijįĢijijĴĩl#ÄwíEĆáĶĻģ
ĤĴĸĺħĬĵ¡įĢģļıĬĨĦĦĬĩĸİ"nù¢0âKēįĥĻĵĵľĩĴ
```

(b)

Fig. 5. Analysis of our proposed Chaotic map for (a) Plaintext (b) Encryption with correct key

Encryption

– Step 14: The values of $ChaosVal$ and ϑ after the map is iterated Itr number of times are as follows:

$$\begin{aligned} ChaosVal &= -1.2054 \\ \vartheta &= ChaosVal \times Itr \\ &= -1.2054 \times 2982 = -3594.4 \end{aligned}$$

– Step 15: The plaintext produced by D_8 at time instant t and (t−1) in our experiment be given as $\mathcal{P}^{t-1} = 10.2$ and $\mathcal{P}^t = 11.6$. The plaintext is encrypted as explained in Sect. 5.4.

– Step 16: The next step is to perform XOR operation of the plaintext $\mathcal{P}^t$, chaotic output ϑ and the previous cipher $\mathcal{C}^{t-1}$ after converting them into their binary 256-bit equivalent as

$$C^t = \begin{cases} \mathcal{C}^t = \mathcal{P}^t \oplus \vartheta \oplus \mathcal{C}^{t-1} & \text{if } t \neq 1 \text{ or,} \\ \mathcal{P}^t \oplus \vartheta & \text{otherwise} \end{cases}$$

– Step 17: Similarly, for integrity checking since the size of $\mathcal{C}^t$ is 256 bits, then the maximum size of N^0 will be $\log_2 256 = 8$ bits.

– Step 18: The resultant $\mathcal{C}^t$ i.e., 256 + 8 bits is converted into their ASCII values generating 32 + 1 characters of ASCII which is the final cipher *Cipher*. Figure 5(a) and (b) shows the plaintext/sensor data corresponding to each of the 8 ciphertexts respectively.

– Step 19: The ciphertext *Cipher* is then sent to TC for decrypting.

Decryption

– Step 20: TC, on receiving the ciphertext *Cipher*, now performs the reverse operation by first estimating the value $\mathcal{K}$ from the information provided by the device D_8.

- Step 21: Value of Iteration Itr and is calculated using similar approach with the help of the pre-shared session key $\mathcal{S}$. Ultimately ϑ is calculated through iteration on the chaotic map with the help of the control parameters, i.e., $\mathcal{K}$, $\mathcal{S}$ and Itr.
- Step 22: TC extracts the plaintext by performing XOR operation of the previous known cipher C_{i-1}, the current cipher value C_i and the output produced by the chaotic map ϑ.
- Step 23: The value obtained after the XOR operation is the plaintext $\mathcal{P}$ after converting to its decimal form, i.e., the value of 10.2 is successfully decrypted by the TC.

Secure Data Fusion

- Step 24: TC individually decrypts the values from each of the devices. After the Ciphertexts $C_1^t, C_2^t, \ldots, C_8^t$ from devices $D_1, D_2, \ldots, D_8$ respectively are successfully decrypted into the plaintexts $\mathcal{P}_1^t, \mathcal{P}_2^t, \ldots, \mathcal{P}_8^t$, the process of data fusion begins.
- Step 25: In this phase, all the sensor information in the form of plaintexts are fused to form a decision. (Note that the algorithm for data fusion is beyond the scope of this paper and thus is not added in order to avoid complication.)
- Step 26: The decision is conveyed to the concerned authority to the IoT application wirelessly or through the monitoring app.

All the steps above are explained with respect to a single device D_8. Similar procedure is followed by all devices in the network, i.e., $D_i | i = 1 \ldots 8$ each follow through the steps of authentication, key exchange, encryption and decryption.

7 Security Analysis

Accomplishment of Anonymity. Anonymity ensures that even if the attacker A eavesdrops on any ongoing communication, he/she should not be able to detect the identity of either the sender or receiver of the intercepted message, i.e. the identity of a device D_i is completely anonymous. This is achieved in our protocol by the one-way property of Hash functions which is the heart of our work in this paper. Intuitively, a one way function is one which is easy to compute but difficult to invert. Thus, even if the hash proofs of the device D_i are intercepted by the attacker, it is impossible for him/her to extract the identity or the RFID of D_i thus achieving device anonymity.

Accomplishment of the Device Authentication. Since our security protocol is based on the MHT where the values at each node are the hash of RFID tags and the RFID tags uniquely identifies a device, the generated hash values are also unique. In our protocol, any attacker attempting to initiate communication with the TC cannot forge the RFID of the authentic device and thus cannot deliver the accurate proofs. Moreover, even if the attacker in some way intercepts the RFID of the device, it is impossible to get hold of the hash values of all its siblings that constitutes the valid proof. In this way, TC authenticates a legitimate device and prevents unwanted communication from untrusted third parties.

Accomplishment of Data Integrity. To achieve data integrity, we have incorporated a mechanism where the ciphertext includes few bits for integrity checking. Suppose the ciphertext C be 11000110. The number of 0s is 4 or $N^{(0)} = 100$. Then *Cipher* would be $Cipher = 11000110|100$ (where | signifies the division of *Cipher* into C^t and its 0 count $N^{(0)}$). Now if the ciphertext 11000110 were tampered by an attacker to 11000100 by changing the seventh bit to a 0, the value of $N^{(0)}$ being the same, the cipher would then be $Cipher = 11000100|100$. For the *Cipher* to be a valid codeword, the count $N^{(0)} = 100$ would also have to be changed to 101 because we now have 5 0s, not 4. But this requires changing a 0 to a 1, something that is forbidden. If the codeword were changed to 11000110|110 by altering $N^{(0)}$, then C would have to be changed so that it had 6 0's instead of 4. Again, this requires changing a 0 to 1 which is not possible. In this way, our algorithm guarantees data integrity.

Accomplishment of the Data Confidentiality. Data confidentiality is maintained in our protocol as there is no requirement of exchanging keys in the network. As a result, any attempt by an unauthorized user to forge identity is nullified. Therefore, the entire process of our proposed architecture is highly confidential and the exchanged data is highly secured against tampering.

Resistance to Replay Attacks. Our proposed is resilient to replay attacks by using a random value of $\mathcal{S}$. In this way, an attacker cannot replay the same message again and again with the intention of passing the authentication phase. Others key parameters such as $\mathcal{K}$ and Itr are not shared in the insecure channel and thus they cannot be intercepted by the attacker. These keys are generated at both ends separately through the control parameters, thus making our proposed scheme secure against from unwanted replay attacks.

Resistance to Forgery Attacks. An attacker may also attempt to use the RFID of any legal validated device to pass the verification process of the TC. In that case, the attacker needs to construct a valid request message REQ with valid proofs to pass the TC's verification. However, to do that, he/she needs to not only know the hash of the RFID but in addition the individual hash values of all the sibling nodes in its path to the TC i.e. apart from $H(RFID)$, other proofs that includes the ϕ values calculated for every node j at level i as $\phi_{i,j} = H(\phi_{2i-1,j-1}||\phi_{2i,j-1})$, which is quite impossible for him/her to figure out as these are the unknown secrets and therefore an attacker cannot convince TC of its identity. In this way, our proposed scheme can resist forgery attacks.

8 Conclusion

Owing to the urgent need for developing security algorithms for Internet of Things (IoT) environment, this paper presents a security protocol by combining the advantages of both Merkle Hash Tree and Chaotic Cryptography. Our contribution is two-fold. First, we develop an authentication protocol based on Merkle Hash Tree that we have improved to suit to an IoT application by utilizing the RFID tags for generating the tree. Secondly, we have designed an encryption algorithm inspired by the chaos theory in cryptography. Additionally, we have proposed a novel chaotic map that has been used for designing the encryption algorithm. The proposed security protocol use lightweight computations that is well suited for the resource-constrained IoT devices. Experimental and security analysis proves the effectiveness of our algorithms and its resilience to security attacks.

Acknowledgement. This work is supported by the Ministry of Electronics & Information Technology (MeitY), Government of India under the Visvesvaraya PhD Scheme for Electronics & IT (PhD-PLA/4(71)/2015-16).

References

1. Guo, B., Zhang, D., Yu, Z., Liang, Y., Wang, Z., Zhou, X.: From the Internet of Things to embedded intelligence. World Wide Web **16**(4), 399–420 (2013)
2. Satyadevan, S., Kalarickal, B.S., Jinesh, M.K.: Security, trust and implementation limitations of prominent IoT platforms. In: Satapathy, S.C., Biswal, B.N., Udgata, S.K., Mandal, J.K. (eds.) Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014. AISC, vol. 328, pp. 85–95. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-12012-6_10
3. Weber, R.H.: Internet of Things-new security and privacy challenges. Comput. Law Secur. Rev. **26**(1), 23–30 (2010)
4. Lampropoulos, K., Denazis, S.: Identity management directions in future Internet. IEEE Commun. Mag. **49**(12), 74–83 (2011)
5. Suhardi, R.A.: A survey of security aspects for Internet of Things in healthcare. In: Kim, K., Joukov, N. (eds.) Information Science and Applications (ICISA) 2016. Lecture Notes in Electrical Engineering, vol. 376. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-0557-2_117
6. Alasmari, S., Anwar, M.: Security & privacy challenges in IoT-based health cloud. In: 2016 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 198–201. IEEE (2016)
7. Islam, S.R., Kwak, D., Kabir, M.H., Hossain, M., Kwak, K.-S.: The Internet of Things for health care: a comprehensive survey. IEEE Access **3**, 678–708 (2015)
8. Roman, R., Najera, P., Lopez, J.: Securing the Internet of Things. Computer **44**(9), 51–58 (2011)
9. Kalra, S., Sood, S.K.: Secure authentication scheme for IoT and cloud servers. Pervasive Mob. Comput. **24**, 210–223 (2015)
10. Amin, R., Kumar, N., Biswas, G., Iqbal, R., Chang, V.: A light weight authentication protocol for IoT-enabled devices in distributed cloud computing environment. Future Gener. Comput. Syst. **78**, 1005–1019 (2018)

11. Zhou, L., Li, X., Yeh, K.-H., Su, C., Chiu, W.: Lightweight IoT-based authentication scheme in cloud computing circumstance. Future Gener. Comput. Syst. **91**, 244–251 (2019)
12. Mookherji, S., Sankaranarayanan, S.: Traffic data classification for security in IoT-based road signaling system. In: Nayak, J., Abraham, A., Krishna, B.M., Chandra Sekhar, G.T., Das, A.K. (eds.) Soft Computing in Data Analytics. AISC, vol. 758, pp. 589–599. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-0514-6_57
13. Nesa, N., Ghosh, T., Banerjee, I.: Outlier detection in sensed data using statistical learning models for IoT. In: 2018 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6. IEEE (2018)
14. Nesa, N., Ghosh, T., Banerjee, I.: Non-parametric sequence-based learning approach for outlier detection in IoT. Future Gener. Comput. Syst. **82**, 412–421 (2018)
15. Rathore, H., Jha, S.: Bio-inspired machine learning based wireless sensor network security. In: 2013 World Congress on Nature and Biologically Inspired Computing, pp.140–146. IEEE (2013)
16. Bodei, C., Chessa, S., Galletta, L.: Measuring security in IoT communications. Theor. Comput. Sci. **764**, 100–124 (2019)
17. Bodei, C., Degano, P., Ferrari, G.-L., Galletta, L.: Where do your iot ingredients come from? In: Lluch Lafuente, A., Proença, J. (eds.) COORDINATION 2016. LNCS, vol. 9686, pp. 35–50. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39519-7_3
18. Nesa, N., Ghosh, T., Banerjee, I.: Design of a chaos-based encryption scheme for sensor data using a novel logarithmic chaotic map. J. Inf. Secur. Appl. **47**, 320–328 (2019)
19. Shukla, P.K., Khare, A., Rizvi, M.A., Stalin, S., Kumar, S.: Applied cryptography using chaos function for fast digital logic-based systems in ubiquitous computing. Entropy **17**(3), 1387–1410 (2015)
20. Wang, W., et al.: An encryption algorithm based on combined chaos in body area networks (2017). http://www.sciencedirect.com/science/article/pii/S0045790617324138
21. Hamad, N., Rahman, M., Islam, S.: Novel remote authentication protocol using heart-signals with chaos cryptography, In: International Conference on Informatics, Health & Technology (ICIHT), pp. 1–7. IEEE (2017)
22. Ning, H., Liu, H., Yang, L.T.: Aggregated-proof based hierarchical authentication scheme for the Internet of Things. IEEE Trans. Parallel Distrib. Syst. **26**(3), 657–667 (2015)
23. Liu, J., Su, H., Ma, Y., Wang, G., Wang, Y., Zhang, K.: Chaos characteristics and least squares support vector machines based online pipeline small leakages detection. Chaos, Solitons Fractals **91**, 656–669 (2016)
24. Furquim, G., Pessin, G., Faiçal, B.S., Mendiondo, E.M., Ueyama, J.: Improving the accuracy of a flood forecasting model by means of machine learning and chaos theory. Neural Comput. Appl. **27**(5), 1129–1141 (2016)
25. Furquim, G., Mello, R., Pessin, G., Faiçal, B.S., Mendiondo, E.M., Ueyama, J.: An accurate flood forecasting model using wireless sensor networks and chaos theory: a case study with real WSN deployment in Brazil. In: Mladenov, V., Jayne, C., Iliadis, L. (eds.) EANN 2014. CCIS, vol. 459, pp. 92–102. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11071-4_9
26. Yang, L., Fei, L.Y., Dong, Y.X., Yan, H.: Iris recognition system based on chaos encryption. In: 2010 International Conference on Computer Design and Applications (ICCDA), vol. 1, pp. V1–537. IEEE (2010)

27. Liu, X., Fang, X., Qin, Z., Ye, C., Xie, M.: A short-term forecasting algorithm for network traffic based on chaos theory and SVM. J. Netw. Syst. Manage. **19**(4), 427–447 (2011)
28. Li, H., Lu, R., Zhou, L., Yang, B., Shen, X.: An efficient merkle-tree-based authentication scheme for smart grid. IEEE Syst. J. **8**(2), 655–663 (2014)
29. Li, D., Aung, Z., Williams, J.R., Sanchez, A.: Efficient authentication scheme for data aggregation in smart grid with fault tolerance and fault diagnosis. In: 2012 IEEE PES Innovative Smart Grid Technologies (ISGT), pp. 1–8. IEEE (2012)
30. Nicanfar, H., Jokar, P., Leung, V.C.: Smart grid authentication and key management for unicast and multicast communications. In: 2011 IEEE PES Innovative Smart Grid Technologies, pp. 1–8. IEEE (2011)
31. Xu, K., Ma, X., Liu, C.: A hash tree based authentication scheme in SIP applications. In: IEEE International Conference on Communications, 2008. ICC 2008, pp. 1510–1514. IEEE (2008)
32. Liu, C., Ranjan, R., Yang, C., Zhang, X., Wang, L., Chen, J.: MuR-DPA: top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud. IEEE Trans. Comput. **64**(9), 2609–2622 (2015)
33. Zhang, H., Tu, T., et al.: Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated Merkle hash tree. IEEE Trans. Serv. Comput. **PP**(99), 1 (2017)
34. Garg, N., Bawa, S.: RITS-MHT: relative indexed and time stamped Merkle hash tree based data auditing protocol for cloud computing. J. Netw. Comput. Appl. **84**(Supplement C), 1–13 (2017). http://www.sciencedirect.com/science/article/pii/S1084804517300668
35. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) Advances in Cryptology — CRYPTO 1989 Proceedings. CRYPTO 1989. Lecture Notes in Computer Science, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21
36. Moreira, F.J.S.: Chaotic dynamics of quadratic maps. IMPA (1993)
37. Lawande, Q., Ivan, B., Dhodapkar, S.: Chaos based cryptography: a new approach to secure communications, vol. 258, no. 258. BARC newsletter (2005)