# Chapter 6
# Lightweight Multi-level authentication scheme for Multi-level IoT-Fog Architecture

In recent times, there is a need for a system that consists of connected devices that can function smoothly, remotely, and such a system is called the Internet of Things (IoT). In simple terms Internet of things is a network that connects devices like (as simple as) mobile phone, television, watches, refrigerators, and other electronic devices like smart cars, air conditioner, room lights, virtual assistant AI technological devices like Alexa, Siri, Cortana, google assistant, etc. to a centralized internet. This connection supports interoperable communication between such devices [121]. These devices can sense and generate multi-volume data, which if sent on the clouds, can overload the clouds and waste the network bandwidth. To make faster decisions, an unprecedented and innovative solution called Fog Computing is introduced that provides advanced techniques such as distributed storage, embedded data mining, and notification service at the edge of the network. Fog Computing integrates Cloud Computing into the network and asserts unsupported fundamentals in Cloud Computing [122]. Some of the fundamental characteristics of Fog Computing are low latency, location awareness, support for on-line analytics, geographical distribution, edge location, interoperability [123]. By the end of every year, more than 50 million devices will be connected on the Internet, which calls for the support of an efficient data storage techniques. To address this challenge, edge computers were introduced using Cloud Computing devices to store local data [124]. However, due to limited resources, it led to resource contention that increased latency. Furthermore, fog computers provide Cloud Computing integration with the network and integrate the edge devices with cloud resources. Fog Computing avoids resource contention, thereby making sure that the latency is not high due to which it is considered an efficient solution to the challenges offered by IoT.

As IoT is concerned with many data and devices, it poses a significant threat for attackers or hackers. Since in IoT myriad devices are connected and it is interoperable, even if one device is attacked, the entire network might fail and the attacker can gain access to a user's sensitive data like passcodes, Credit Card number, details of financial accounts, location, etc. [125]. Authentication is a significant security issue in Fog Computing. It provides services to many end-users; wireless network security is a big issue since wireless is predominant in Fog Computing, and thus attacks like jamming attacks and sniffer attacks could occur [126].

A multitude of cryptographic algorithms can be used for IoT devices' security, but because it provides limited resources like limited memory, insubstantial RAM, limited power, it becomes difficult to use such cryptographic algorithms intensively [127]. Thus, there is a need for a lightweight cryptography algorithm that can offer security efficiently to IoT devices without taking up a significant load. Effective implementation of IoT demands low latency and small chip area occupied on hardware or memory requirements for the software execution. Lightweight encryption supports both these requirements, and thus it compliments IoT, making it the best choice of security algorithm for IoT [127].

The proposed scenario illustrates the need for Fog Nodes at different levels and how they communicate with one another with the help of horizontal and vertical communication. Real-life scenarios are shown to demonstrate how lightweight encryption algorithms could be used at different layers to ensure that the security is preserved. Having multiple security algorithms make it more difficult to deploy any application, while the high complexity because of the multiple security schemes makes the system more vulnerable. So here, one unique lightweight security scheme is suggested which can single-handedly satisfy all the levels of security with different security levels. The scheme is based on Logical operations and it is multi-keyed logic.

The proposed scheme makes sure that it has a lower time and space complexity to support IoT devices. In addition to that, the scheme has varied key-length feature which makes it capable to secure higher-level data at higher-level devices like servers. In designing, different HMACs, Op_codes, and chaining logics are used. The proposed

algorithm is also compared with other existing algorithms and it is discovered to be superior to the others. Furthermore, it is tested against various attacks to prevent any vulnerabilities from the exploitations. The given security scheme is lightweight and it provides security assurance in data transmission in the IoT-Fog, Fog to multi Fog, and Fog to Cloud end.

This chapter starts by discussing the traditional security measures available and it focuses on lightweight encryption and its features. Further it deals with the research gaps and the need for a multi-level Fog Architecture which is proposed. A lightweight encryption algorithm is proposed to satisfy the need of the levels in the architecture. Evaluation and comparison with other available algorithms is discussed followed by the concluding remark.

## 6.1 Traditional Security Approaches

Different traditional security approaches are also introduced here to provide a better insight into the security algorithms.

**1) Cryptographic technique.**

The cryptographic techniques which comprise the symmetric key algorithms like Advance encryption standard (AES) and asymmetric algorithms like Rivest Shamir Adelman (RSA) cannot be used for the security of the IoT applications [128]. The higher CPU configurations required by these algorithms make them expensive for their implementation in this area. Hence a new need of creating more feasible security algorithms has come up. The symmetric key algorithms use the same key for encryption and decryption at the sender and the receiver's end while the asymmetric algorithm uses different keys for encryption and decryption. The sender encrypts the data to be sent on the network using the public key and sends it in the form of ciphertext, and the receiver decrypts this ciphertext using his private key. The asymmetric algorithms require appropriate key management as multiple keys are involved and become complex to implement compared to symmetric key algorithms where only a single key is engaged.

**2) Key management.**

The most vital point of implementation of any cryptographic algorithm is key management. In the IoT architecture, the raft number of connected nodes in an IoT network and the myriad data hinder the existing key management techniques [129-131].

**3) Denial of Service.**

The Denial of Service (DoS) attack launches many requests to the system and makes the system unavailable for serving actual requests. The IoT applications in the healthcare domain can lead to loss of lives in case a DoS attack is launched on them. The battery-operated IoT devices prevent DoS detection on the sensor nodes as even sample attack messages cannot be found on the IoT devices to detect and occlude them [132].

**4) Authentication and Access Control.**

It is feasible to implement Authentication of devices before the communication of data between a limited number of devices with protocols like SSL handshake. As far as IoT is concerned, many devices are involved in data communication amongst themselves, so Authentication of data using these existing protocols is not feasible. Thus, more research must be done to fulfill the authentication and access control of the nodes within an IoT network.
These traditional security approaches are not suitable to secure IoT data due to the different time and space complexities of the IoT context. The solution to this problem is we need Light Weight encryption techniques.

## 6.2 Lightweight Encryption:

Lightweight encryption's primary focus is to optimize the cryptographic algorithms based on standard cryptographic primitives to run on small and resource-constrained devices. The aim is to provide the Authentication and encryption in one pass by

ensuring the communicating entities that their information is not tampered with. The IoT devices require less-intensive computational resources and lower power consumption due to the utilization of battery.
The features of lightweight encryption are –

**1. Speed**

The set of instructions can execute faster and hence provide the results at a much faster rate. This would benefit from getting the insights into the data quickly as the sensors capture the raw data.

**2. Power Consumption**

The execution of the set of instructions takes place faster, and hence the system can return into an idle mode as quickly as possible to minimize the power utilization. This helps the IoT devices to function more efficiently as they are battery operated and need minimum power utilization.

**3. Computation**

The Lightweight encryption is supposed to run on the IoT devices which handles smaller data but greater in numbers. So, this encryption scheme should take minimum computation power, as IoT devices have limited computing capacity. Good Speed, less power consumption, and fast computation can be achieved in lightweight encryption by using different logical operations [133] and various combinations of security keys like multiple keys, chaining mechanisms, and other functions to achieve non-repudiations. [134]

## 6.3 Research Gap

Based on the detailed literature review done in the chapter 2. The observed research gaps are listed here

→ IoT devices remain unsupervised for a longer duration of time, so they need the data and access security protocols.

→ In specific scenarios, IoT-IoT devices also need to be communicated hence it should have data sharing security.
→ By considering the computational capabilities of the IoT device, a lightweight encryption scheme is needed.
→ Single Fog Nodes may lag in terms of computation and decision-making time, so we need multiple Fog Nodes at the same horizontal levels to support it.
→ We need an efficient security scheme for sharing data across multiple Fog Nodes at the same and at different levels to achieve parallel and Distributed Computing among the Fog Nodes to facilitate big data analytics.
→ To reduce the cloud traffic, the multilevel Fog Node scenario is suggested, and also its security solutions are provided.
→ For multiple levels, we need multiple security algorithms which makes the system more complex and vulnerable because every Algorithm is covering one or the other aspect of the security or computation.

A unique style lightweight IoT-Fog context-aware security scheme is suggested to satisfy all the security needs at all the levels is shown in figure 6.1.

## 6.4 Proposed System Architecture and System Designing

The literature survey shows that several challenges are needed to be addressed. A multi-level multi-Fog scenario is needed to deal with large amounts of data that needs a lot of processing. Horizontal and vertical communication between underlying Fog Nodes is necessary to guarantee more security, low latency, and more encryption as the level increases. The Research Gaps stated above shows the importance of lightweight encryption scheme and lightweight encryption algorithms. Based on the gaps and surveys conducted, a System Architecture is proposed to deal with the challenges faced.

### 6.4.1 System Architecture

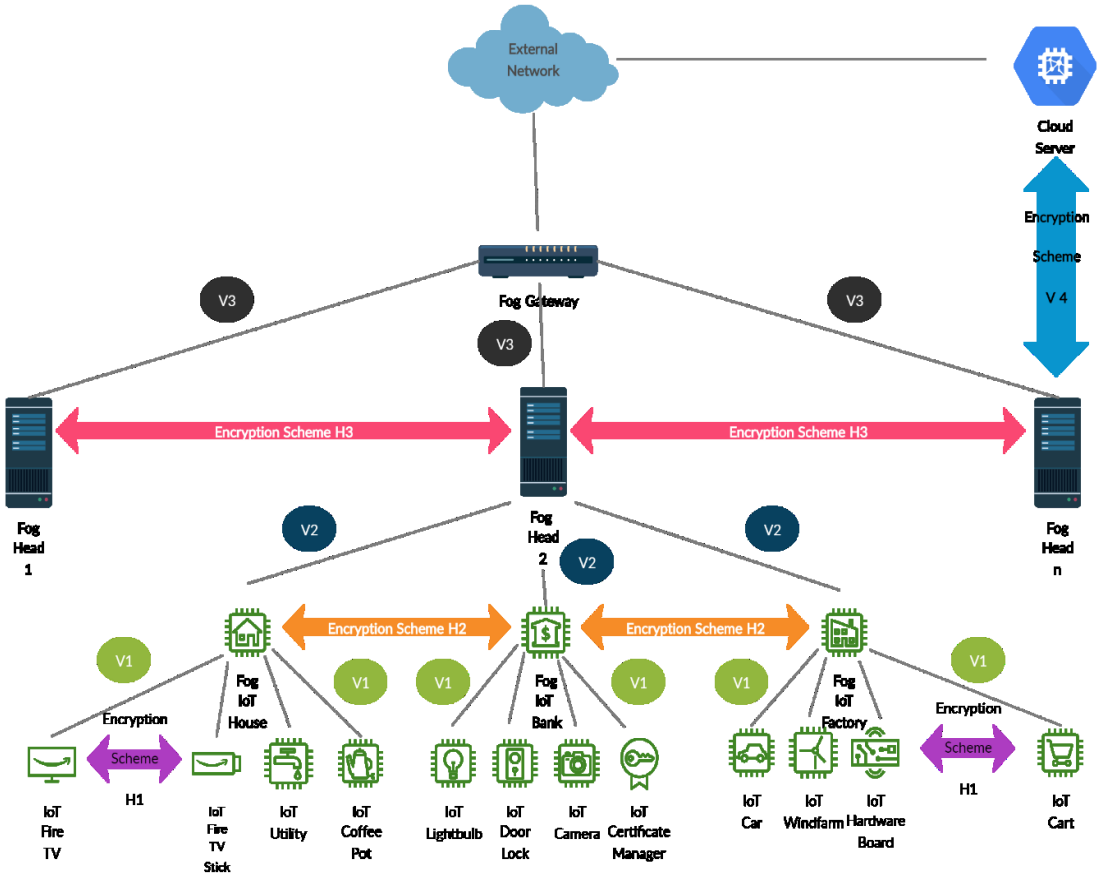


**Figure 6.1** Multilevel Fog Nodes scenario with IoT and Cloud Servers

**System description**

The following diagram exhibits a future possible real-world scenario where the Lightweight Encryption algorithm is vital to use.

In the diagram H1, H2 and H3 denote Horizontal Encryption Scheme, whereas V1, V2, V3, V4 denotes Vertical Encryption Scheme. It is important to note that as we move in a bottom to the top manner, the encryption scheme becomes more and more secured due to the significance of the data in that respective Node, which means that V1 will have a more robust encryption as compared to H1. H denotes the security scheme at horizontal levels while V represents the vertical security levels.

**Fog-IoT Bank :**

The Fog IoT Bank consists of several devices which include the IoT Lightbulb, IoT Door Lock, IoT Camera, and IoT Certificate Manager. These devices will be mainly

used to enhance the security of the depositor's money and the private data about the Bank's customers like account number, transactions taking place in that account. Every Bank must guarantee the safety of its customers' information and their money, thereby should be aware of internal and external threats. Thus, the need for these measures is increasing more than ever. One such scenario can be in the Bank's locker room, which contains the depositor's money and the servers containing valuable information about the Bank's customers. If the IoT-powered door is opened by any false means or the lock is broken down by anyone who does not have access to the bank locker room, then the door lock will alert the Fog Node using V1. The Manager or any Higher Authority now has access to close the door to ensure that the intruder does not escape the room. If the IoT powered Camera found out that suspicious activity is happening in the locker room, it will notify the Fog Node. The footage recorded by the Camera can aid the Bank and the investigative authorities to take strict action against the intruder. Right after the intruder enters the room, the IoT Lightbulb will switch on to indicate the possibility of any theft in the locker room and will give the alarm signals to the Fog Bank node via V1. Now, suppose the intruder tries to tamper with the customer's data present on the server. In that case, the IoT Certificate Manager will immediately encrypt all the data present on the server. Thus, the invader would not be able to access the customer's data. If fog IoT bank is compromised and does not respond to take appropriate actions in that case IoT door lock will use H1 encryption scheme to alert other IoT devices present in the vicinity like IoT light bulb, IoT camera, and the IoT Certificate Manager.

**Fog-IoT House :**

The Fog IoT House consists of multiple devices that can be controlled by electronically connected devices or internet-based systems. Consider a scenario of an IoT-enabled house with numerous devices such as IoT Fire TV, IoT Fire TV Stick, IoT Utility, IoT Coffee Pot. The IoT Fire TV is a smart television that works on the Internet and gives the user, access to a multitude of applications using the Firestick. If the TV heats up to a temperature beyond its threshold level or any power fluctuations taking place, the IoT TV will send signals via V1 to the House's Fog Node to immediately shut down the TV. But due to any circumstances if the Fog Node does not take the appropriate action within a stipulated time, then the IoT Fire TV stick will take the control in hand, and

with the help of Horizontal Scheme H1 give the command to the TV to shut down, thereby preventing any damage to the TV. IoT Utility can be configured for many applications like maintaining the quality of water in our House and detecting if the water contains any deviations from the norm, such as arsenic or radium which a noxious for our health. If the IoT Utility detects that the water is contaminated, it alerts the Fog Node not to consume it. The IoT Coffee Pot is a device that can be used for several purposes. If the coffee inside the Coffee Pot remains for a prolonged period, it will send signals to the Fog Node stating that the Coffee is not suitable for drinking as the coffee's oils start to go bad, which might alter the taste of the coffee.

**Fog-IoT Factory :**

Fog IoT Factory includes various devices which are IoT Vehicle, IoT Windfarm, IoT Hardware Board, and IoT Cart. We could use these components to know about the Factory's electricity requirement by adding sensors to the devices that send the data to the Fog Node. IoT Vehicle is used to transport the goods that the Factory manufactures, and we could easily deduce the electricity that the company might use by the number of trips that electric vehicle makes to supply the goods manufactured to the distributor, which then sends the data to the Fog Node. IoT Windfarm can communicate with the Factory's Fog Node using V1 Scheme where the Windfarm notifies the Fog Node when the Windfarm might not be able to generate enough electricity that meets the Factory requirements. The Fog Node might then take appropriate actions to arrange the deficit in electricity by using the government's commercialized electricity to meet the electricity demands. Windfarm can also notify the Fog Node if it generates surplus electricity wherein the fog mode then takes appropriate steps to store this electricity. The IoT cart deals with supply chain management, and the IoT Hardware board is responsible for controlling the manufacturing operations. Based on demand and supply, the IoT enabled car, Windfarm, hardware board, and IoT cart will work synchronously.

Fog IoT Factory can communicate with Fog IoT bank and vice versa using the Horizontal encryption scheme H2. Also, Horizontal security schemes can facilitate secured data communications to achieve parallel and Distributed Computing. The Factory might require detailed financial reports about their accounts in the Bank or

might need some Bank financing. Since this data is crucial, we need more encryption in H2 than the data transferred via H1. Another example supporting H1 communication is discussed in the literature review, and it is about controlling the water pump to maintain the soil moisture by using soil moisture sensors.

Furthermore, the communication between the Fog Nodes and the Fog Head takes place via V2. For example, the regulatory authority of a district can act as a Fog Head. It stores information about all the Banks and the Factories in the same district, which can help them make decisions. If the Fog Head got to know that many banks have lax security in protecting the depositor's money, then the authority might give a warning to the Banks to increase their security or impose a fine on the Bank. Also, if the Factories are squandering electricity, then Fog Head might warn the Fog Node of the Factory to use electricity wisely. The Fog Head can even know if any Factory has surplus electricity, then Fog Head might supply it to another Factory which is on a deficit. Thus, Fog Head can play the role of adequately managing the resources that the district possesses.

Communication between different districts can also take place using the Horizontal Scheme H3. Say if a district 'A' having Fog Head 1 possess an excess of electricity then district 'B' having Fog Head 2 which is currently facing a shortage of electricity can communicate with district 'A' to supply the excess electricity thereby meeting the electricity demands of the district 'B'. If a Bank in district 'A' has an excess of cash, it might lend it to some other Bank in district 'B' that is currently facing a cash crunch.
Consider the Fog Gateway as a state controlling operations of various districts present in the state. Communication between a specific district and the state takes place with Vertical Communication V3, as shown in the figure above. Say if a district 'A' with Fog Head 1 needs more power due to the overall development of the district, it will directly communicate with Fog Gateway, the state authority. This ensures that the needs are satisfied. Say if a district 'B' with Fog Head 2 needs more funds to be allocated due to upcoming projects in the district, vertical communication occurs between the district and the state. Or the district-level Fog Nodes will communicate to the state level Fog Nodes using the V3 communication scheme, and once the communication keys are set up, then the further direct communication between two district-level Fog Nodes can take place using H3 level.

Now consider the Cloud Server, which consists of all the districts' data and the states of a nation. Data can be directly fetched from the Cloud Server using Vertical Communication V4 by a specific Fog Head. Consider a case when a district' N' wants statistical data or analysis about the supply of medicines, foods, funds or electricity, or any other resources then it will directly interact with the Cloud Server instead of the Fog Gateway. This takes place with the help of the Vertical Communication V4. This takes place because the Fog Gateway consists of data from many districts, and it isn't easy to find data from a particular district. Hence the district' N' communicates with the Cloud Server and fetches the data or the detailed analysis that it requires.

The data transferred via V4 should be more encrypted than the data transmitted via V3 because the information stored in the Cloud Server is more valuable as it contains information about all the states of a nation.

## 6.5 Designing the secured scheme

To make the Algorithm complete in all security views, Op_codes, HMACs, timestamps, symmetric, and Asymmetric Algorithm approaches are used.

### 6.5.1 Opcodes

Op_codes are the Operational codes used in the system for communication. It helps understand the system status. These Op_codes help the sender and the receiver identify the message context, format, and help prevent session hijacking attacks as without the proper sequence of Op_codes, the device will not allow any communication message to go further. The receiver behaves as per the received Op_codes. The set of Op_codes are defined in table 14.

**Table 14:** Defined Op_codes

| Op_code | Authentication Process |
|---|---|
| 1 | Authentication Request |

| | |
|---|---|
| 2 | Authentication Confirmation by the sender |
| 3 | Authentication Reply |
| 4 | Authentication Confirmation by the receiver |
| 5 | Sending Data |
| 6 | Acknowledgment for Received Data |
| 7 | Report for the Error Messages |
| 8 | Sending Configuration Messages |
| 9 | Sending Data Messages |

### 6.5.2 System Functions and Parameters

1) $MAC_i$: MAC address of the $i^{th}$ device.

2) g,p: two numbers present at the End Devices at H1 level, calling them as Devices D1 and D2

3) a,b: are the unique numbers present only at end devices D1 and D2 respectively.

4) K: security key derived by $g^c$ mod p, where c is a or b

5) $T_i$: Timestamp of the $i^{th}$ device

6) $A_i$, $B_i$: $g^c$ mod p, where c is a or b pre-existing with the device.

7) X $+_2$ Y: Logical OR of X and Y

8) X $\oplus$ Y: Logical Ex-OR of X and Y

9) HMAC(): Hash Message Authentication Code of a given value

10) En() : Simple Encryption Function by logical XORing, passed parameters

11) $K_{PU}$: Public Key

12) $K_{PR}$: Private Key

13) En_RSA(data,ψ): Encryption by RSA Algorithm using ψ key

14) De_RSA(data,ψ): Decryption by RSA Algorithm using ψ key

15) $P_{PR}$: End device Private Key.

16) En(M, N): Encrypting M value by using N key by logical XOR

17) Dafun(a,b,c) = ( a $\oplus$ b) + c = z

18) IDafun(a,b,c) is used to retrieve the value of a from the received value, provided b and c is existing. The calculation is as given

a = IDafun(a,b,c) = ( z – c ) $\oplus$ b  if x >= c otherwise

= ( z + comp(c) + 1) $\oplus$ b  if z <c

19) comp(c) = Logical bitwise complement of the c

20) p and q: p is prime and q is co-prime of p

21) $e_i$, $d_i$, and $N_i$: These are dataset for the RSA keys

22) $K_i$: user authentication keys

23) $AuK_1$to $AuK_5$: Authentication Keys

24) ΔT: Timestamp difference

25) $SR_1$ to $SR_3$ and $SBR_1$ to $SBR_2$: Private Authentication keys on the client and server-side respectively

26) $P_{Cj}$ : = $q^{SRj}$ mod p, 1<=j<=3, Client Public Keys

27) $P_{Sj}$ : = $q^{SBRj}$ mod p, 1<=j<=2, Server Public Keys

28) $DyK_{1-12}$, $DyA_{1-4}$, $DyB_{1-4}$ : Dynamic Keys

29) $SK_j$ : $(P_{SRj})^{SR_j}$ mod p, 1<=j<=2

30) $TK_{0-n}$ and $NTK_{0-n}$: these are the set of keys used for encryption and decryption purposes where n is the total number of desired keys based on the values of i and j.

31) RSA_En($M_i$) : $(M_i)^{e_i}$ mod $N_i$: $M_i$ is the plain text in the BigInteger form

32) RSA_De($C_i$) : $(C_i)^{d_i}$ mod $N_i$: $C_i$ is the ciphertext in the BigInteger form

33) $SK_j$ : $(P_{sj})^{SBRj}$ mod p, 1<=j<=2 : secret keys

### 6.5.3 H1 Security Scheme:

Initially, Device D1 will have g, p, $MAC_1$, $MAC_2$, and a. The device D2 has g, p, $MAC_1$, $MAC_2$, and b. The Authentication between two devices takes place by following the sequence given below.

**Step 1:** The device D1 will initiate the communication process by setting up the Op_code value as 1, as it is the authentication request. It calculates $A_1$, and the Message format is as follows.

Op_code | $T_1$ | $A_1$ | HMAC( $MAC_1$ +$_2$ A)

**Step 2:** After receiving the message from step 1, the receiver device D2 will read the Op_code as step 1, and it understands the authentication request by calculating B and the key-value K. the Op_code value is 2 in this case as it is authentication reply by D2 to D1. The message format is as follows.

Op_code | $T_2$ | $B_2$ | HMAC( $MAC_2$ +$_2$ B)

After receiving the message formatted as in step 2, the D1 will calculate K. By using the Key K, D1 and D2 can now have the secured communication.

**Step 3:** The syntax for further data and the acknowledgment communication are specified in step 3 and step 4. The Op_code value is 5 for the data sending purposes.

Op_code | $T_1$ | En(Data $\oplus$ Key) | HMAC( Data +$_2$ $MAC_1$)

**Step 4:** The D2 device receives the message, and it can retrieve the data by XORing En() value with the key. The receiver can also cross-check the integrity of the data by comparing the received and calculated HMAC values. The D2 will reply to D1 as an acknowledgment of the data as

Op_code | $T_2$ | En( Reply_Data $\oplus$ Key ) | HMAC (Reply_Data $\oplus$ $MAC_2$)

After receiving the message, D1 will check, and if any error is found, it replies to D2 using the error Op_code message.

### 6.5.4 V1 Security Scheme:

In this scenario, the D will be the End Device, and F is the Fog Head node. Here, the D1 is computationally less able than the F1 device. D1 will employ symmetric cryptography while F1 follows Asymmetric.

**Step 1:** Device D will send the first authentication request with the respective Op_code and its current timestamp. The device D will send its private key to the Fog Node F by using F's RSA public key. The format of the first message is as follows.

Op_code | $T_D$ | En($P_{PR}$, $K_{PU}$) | HMAC( $P_{PR} \oplus MAC_D$)

**Step 2:** The Fog Node F is more capable and can do many computations than the device D. After receiving the message from Step 1, the Fog Node F will decrypt the D's private Key. The acknowledgment and verification of the key $P_{PR}$ are given by F to D as below.

Op_code | $T_F$ | Dafun($MAC_F$, $P_{PR}$, $MAC_D$) | HMAC($P_{PR}$ +2 $MAC_F$)

**Step 3:** Device D will use the inverse Data authentication function to retrieve the Fog Node's MAC address. This is an important step to use the Data authentication function, and it is to prevent the devices from following the commands from any other Fog Node. Now divide D has got the Fog mac address, and the Fog Node has got device D's private key $P_{PR}$. Now the data communication will take place between the fog and the device. The data communication steps are as follows.

Op_code | $T_D$ | En_RSA(Data, $K_{PU}$) | HMAC( Data +2 $P_{PR}$)

**Step 4:** the Fog Node can send the data to Device D by using the following message

Op_code | $T_F$ | En(Data, $P_{PR}$) | HMAC( Data $\oplus$ $P_{PR}$)

### 6.5.5 Higher level security schemes

The security scheme from the level H2-V2 onwards is shown in figure 6.2. The application architect can choose the needed level of security, and accordingly, the user will decide the key length used in the encryption and decryption process. The

set of steps are as follows. The procedure starts by selecting the p and its co-prime number q.

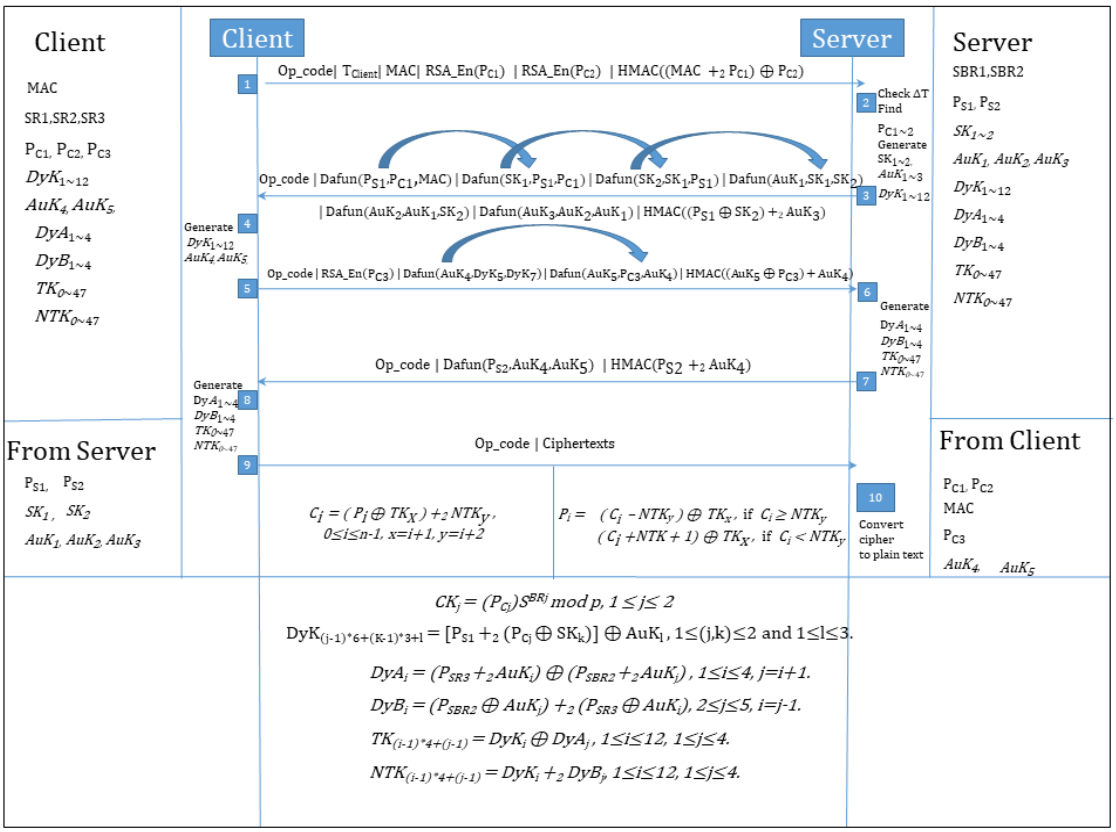


**Figure 6.2** Security algorithm for H2, V2, and higher levels

All encryption and decryption logic is achieved by using the java class called "BigInteger". It supports all the logical steps in the Algorithm and works well with the numbers. If the message is in the form of a number then it works directly in the system. If the generated message is in text form, then one extra conversion step gets added to the system. It is explained in 5.5.6.

### 6.5.6 String to BigInteger and BigInteger to String conversion

Algorithm-1 and Algorithm-2 are describing the logical steps to convert the text to numbers and vice-versa. The pseudo-code is as follows.

**Algorithm -1**

1) start

2) set i=0, BigInt b=0, String s is input

3) if i < string length go to step 4 else go to step 9

4) code= take the ith character and convert to int

5) convert code to BigInt

6) b= do leftshift(b) by 8 bits

7) b=do b OR c

8) i=i+1 goto step 3

9) return b

10) stop

**Algorithm -2**

1) start

2) set String s="", BigInt b is input

3) if b is zero go to step 9 else go to step 4

4) BigInt c= 11111111

5) Int cb= b AND c

6) convert cb to char cv

7) s = concat cv to s

8) b = right-shift b by 8 bits go to step 3

9) return b

10) stop

## 6.6 Implementation results

To test the proposed Algorithm in terms of time, it is subjected to different configuration devices. Three devices are chosen with the following models and configurations. They are Alienware 13 R2 Model, Apple MacBook Air i5 16 GB with Catalina version 10.15.5, and Raspberry Pi model 3. All categories of devices are chosen from very lower to moderate to higher configuration devices. The reason to choose such configuration devices is to mimic the scenario of the end device, fog device, and higher-level Fog server device. And the other reason is to check the platform independent-ness of the proposed security scheme. When the first experimental setups were carried out on the Alienware 13 R2 model and on the

Raspberry Pi device to get the average time taken for RSA key-pair generation, it is recorded and shown in figure 6.3 as follows.

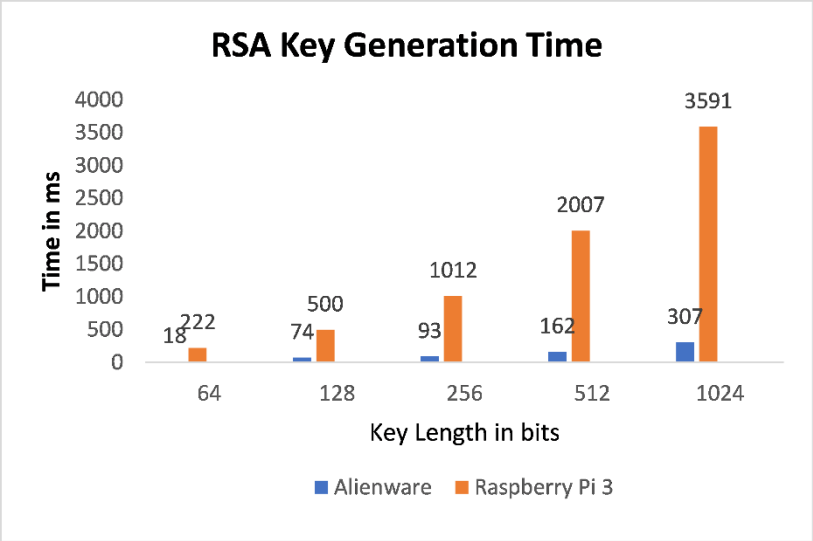


**Figure 6.3** RSA Public-Private Key-pair Generation time

It is observed that the running time is increasing with an increase in the key size. It is because of the prime number length. The prime number is higher-order for higher key lengths, and it takes more time to get its co-prime number. But this process is carried out only once throughout the device lifetime till it remains in the same network with the same neighbouring devices. Setting up the higher-order key lengths also does not matter when it comes to real-time encryption and decryption process for real-time data communication. The higher-level security algorithm on a high configuration device takes only 159 ms from the data encryption process till the data is decrypted.

Further, the text to numbers and numbers to text also plays a vital role in this Algorithm. So, the average time taken for the text to number conversion is as given in figure 6.4. The time taken for the conversion is significantly less. Also, the maximum resolution for the data generated by general-purpose IoT devices is lesser than 10 bytes. So the conversion time for text-based data will always be less than 0.2 ms for high configuration devices, and for low configuration devices, it is always less than 2 ms.

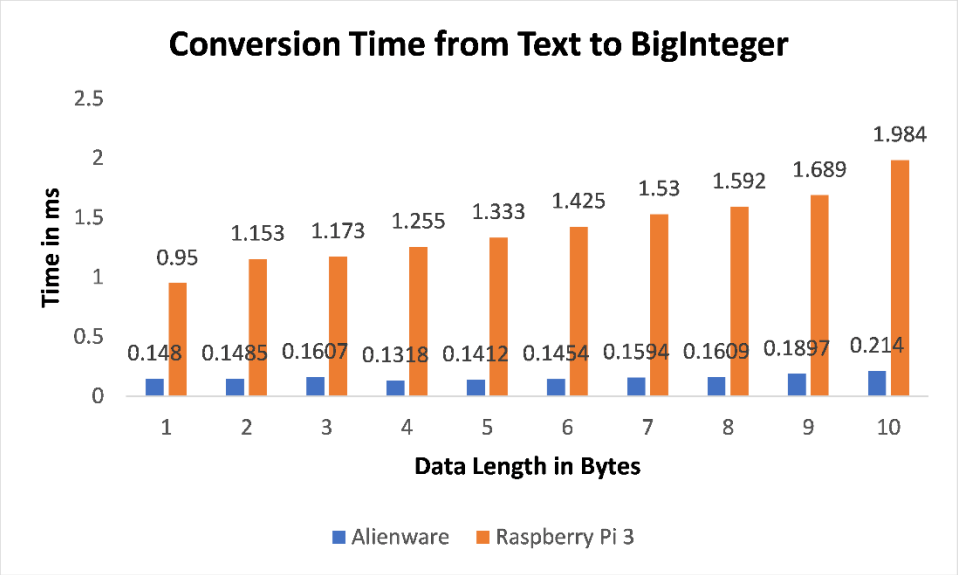


**Figure 6.4** Comparing conversion times of different devices for different data lengths

Further experiments are carried out to find the time and space requirement of the proposed algorithm with other existing algorithms. Few existing algorithms are chosen and they are executed on Apple MacBook Air devices. The time taken in ms by different algorithms are as shown in figure 6.5.

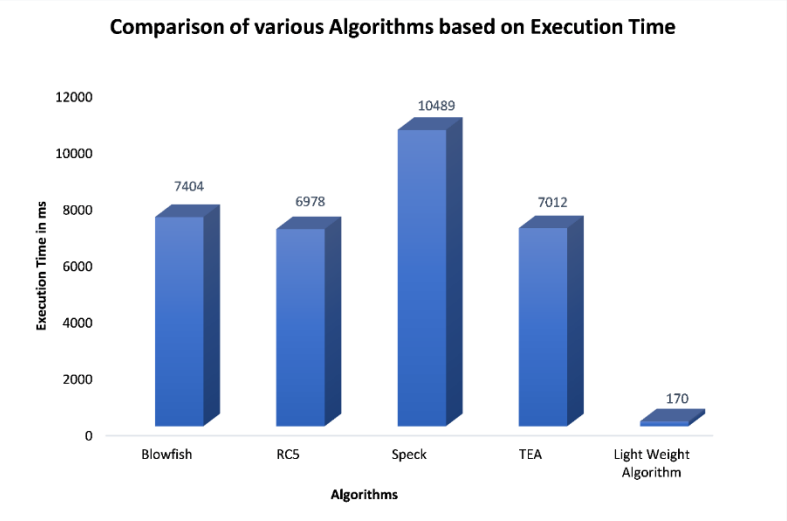


**Figure 6.5** comparing time complexities of various algorithms

In terms of memory requirement, the Lightweight proposed scheme is taking the minimum time. The other algorithms are executed for a 256-bit key length whereas the proposed scheme is executed for 1024 bit key lengths. The time taken by the proposed scheme is very less because it uses logical operation to get avalanche effects

in the ciphertext whereas most other algorithms are using multiple rounds of permutations and combinations with bit shifting mechanism.

The memory requirements of different algorithms are found. The comparison is as shown in figure 6.6. The memory requirement of the proposed scheme is higher than the TEA and BlowFish algorithm. It is because the existing scheme generates multiple keys to use the chaining mechanism and to assure the role of key serializability in the encryption and decryption process. The total memory taken by the proposed algorithm for 1024 bit key length is around 4 MB, which is very much feasible in the lower-level devices. To get it more feasible in much lower level devices the key length can be reduced further. The memory requirement of the proposed system is higher than the TEA and BlowFish but it is manageable at the Fog Node.

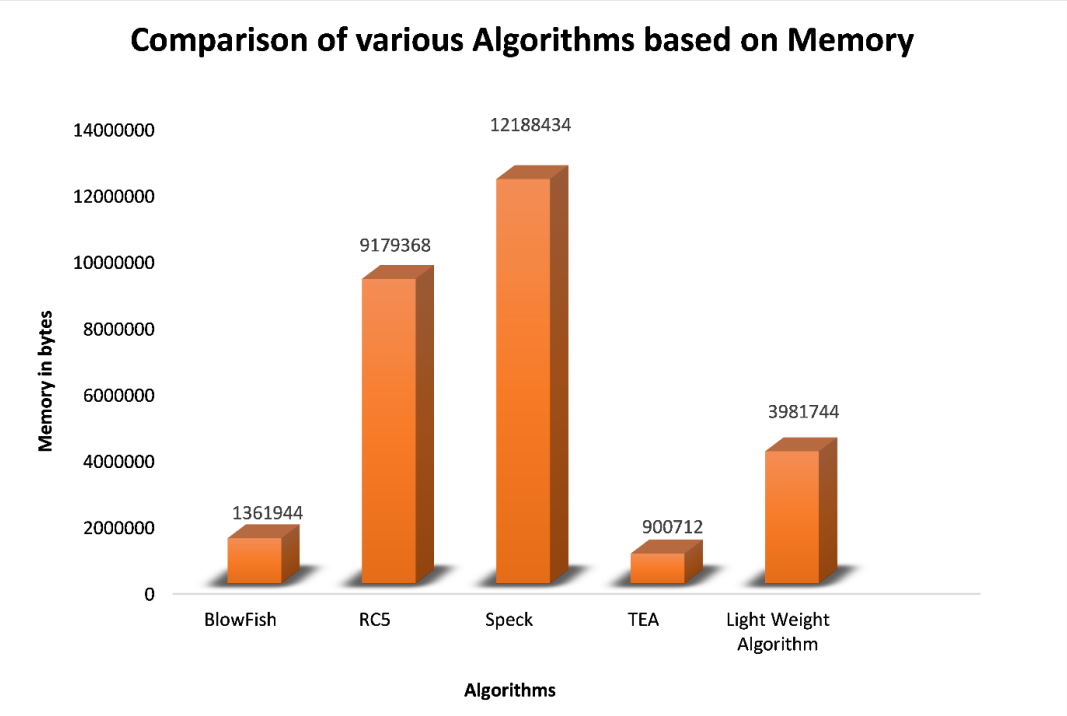


**Figure 6.6** Comparing space complexities of different algorithms

## 6.7 Advantages of the Proposed Algorithm

1) It considers the time stamp value in each message to prevent the replay attack. If the message delivery takes more than the expected value of the network delay, it simply discards the message to avoid the Man-in-the-Middle attack.

2) Every step in the H1 security scheme ensures that the security principles Confidentiality, Integrity, and Availability are satisfied at each step.
3) Based on every device's computational capabilities, the security scheme is assigned symmetric and asymmetric. To make optimal use of it.
4) Initial key sharing step is the most vulnerable step in computing security architecture. We have tried to reduce it as much as possible, and the other parameters are shared based on the pre-existing parameter values.
5) The Hash function used in this scheme is simple, lightweight, logical bit based, and uses simple logical operations.
6) These mentioned security schemes are applicable for any two communication entities, whether they are client-server or peer-to-peer.
7) Minimum sharing of security schemes. And if sharing it is considering Diffie-Hellman approach of sharing security parameters in an unsecured environment.
8) Use multiple security keys and that too in a particular sequence, so even if the attacker gets few keys, it will not be useful without its proper ordering.
9) Use of multiple keys to encrypt and decrypt the multiple data blocks.
10) The Key length can be decided by considering the length of p and q prime numbers. The key length will be twice the prime number p and q's length.
11) The Algorithm also supports the non-standard key length other than 2n power.
12) The suggested security algorithm works for both text and numeric data.
13) The proposed scheme is non-centralized, so even if the central server fails it will work for the pair of communicating devices.

## 6.8 Security against different vulnerable attacks

1) Use of chaining mechanism to strongly support Non-Repudiation.
2) Using random numbers and session-based communication, ensure that the security keys get expired often and hence prevent Replay attacks.
3) The suggested Security scheme is adaptable for different key lengths; hence higher levels algorithms are challenging to get Brute-forced.
4) The Op_code based scheme makes it secured against DoS and DDoS attacks because, without the proper Op_code sequence, the receiver will not entertain the message.

5) Use of different Timestamps make sure that Man-in-the-Middle and Replay attacks are impossible.
6) In the given scheme, few keys are pre-existing on the sender and receiver side, making Spoofing attacks impossible.
7) Every security scheme level is designed for sharing the data in the encrypted, which makes Sniffing attacks impracticable.

**Concluding Remark**

IoT devices remain in remote locations to sense the data and most of the time they are unattended. This further explains the need for security for IoT devices. When any IoT device is communicating to any fog device or the server device, it must make sure that the device it is sending or receiving the data is fabricated in the network to steal the data. Thus, device-to-device authentication is also needed. In addition to this, IoT devices need some lighter version of the security to protect their stateless generated data. IoT data reaches the server via Fog Nodes, fog gateways, and then to cloud gateways. This data, while traveling needs multiple encryption and decryptions. Here, one Light Weight security scheme is suggested which can fulfill the data security need at all the levels in the given scenario. The implemented scheme is based on logical operations like OR, EXOR, and AND. Also, it makes use of some data authentication logic which assists it in averting repudiation attacks. In order to protect against replay attacks, the timestamps and Op_codes are used. The proposed scheme is tested for different categories of devices having different hardware configurations and running different software platforms. The time and memory taken by the suggested algorithm shows that the current scheme is more efficient than the existing algorithms. The initial key sharing in the proposed algorithm is designed in such a way that the initial handshaking in the communication process always remains secure. It also satisfies all three principles of security that is confidentiality by encryption, integrity by using hash functions, and availability by using Op_codes and sequence of ordered keys. Hence the proposed scheme is light-weighted and more secured, and is applicable in IoT-Fog-Cloud communication scenarios.