Hindawi
Journal of Electrical and Computer Engineering
Volume 2019, Article ID 1986325, 12 pages
https://doi.org/10.1155/2019/1986325

Research Article

Low-Cost and Secure Communication System for SCADA System of Remote Microgrids

Amjad Iqbal and M. Tariq Iqbal

Faculty of Engineering and Applied Sciences, Memorial University of Newfoundland, St. John's, NL, Canada

Correspondence should be addressed to M. Tariq Iqbal; tariq@mun.ca

Received 8 February 2019; Accepted 2 April 2019; Published 23 May 2019

Academic Editor: Nicola Sorrentino

Renewable energy-based local microgrids are gaining popularity despite the unavailability of low-cost, power efficient, and secure communication system for its supervisory control and data acquisition (SCADA) system. This research has been carried out to address this issue along with the additional features such as data uploading to a server through a gateway, local data logging, and alerting the concerned crew in case of any fault to minimize the outage time. This paper presents the design of a communication system for the SCADA system of microgrid. ESP32 with LoRa has been used for communication between two nodes or a node and central SCADA unit. Communication security has been achieved by implementing AES cryptography. Data authenticity has been achieved by introducing a unique message authentication code (MAC) for each message. A mesh-like network has been implemented to improve the LoRa range. ESP32 and dragino-uno based LoRa gateways have been tried for uploading the data to the server, and local data storage has been achieved using an SD card. The main controller working as the SCADA unit has the feature of sending emails. Detailed system design and test results are presented in this paper.

1. Introduction

In Canada, more than 300 communities are in remote areas and are isolated from the central power grid and only rely on distributed power generation units. Few of these communities, e.g., Ramea, NL, have added renewable energy to its energy mix. To ensure the power quality and control of these distributed power generation sources, the SCADA system becomes an integral part of the microgrid network. In the smart grids (SG) and the microgrid network, the main challenges for the SCADA system are the lack of low-cost, secure, and authentic communication systems with minimum power consumption [1–3]. In 2008, the Russian Army took charge of the Georgian electric grid by controlling the SCADA system of their grid. This made them realize the importance of the security of the communication system for the microgrid. According to the Wall Street Journal report, in 2009, spies hacked the control system of the U.S. electrical grid and disrupted the system [4]. Further, a number of other articles like [3] have highlighted the concerns about smart-grid cyber security. Therefore, the communication system of an electrical grid, specifically, the setup related to the SCADA system, must have strong resistance against eavesdroppers and masqueraders. Usually, a communication system is regarded as secure if it satisfies the following four features [5, 6].

1.1. Privacy. The message should be encoded or encrypted such that only the authorized receiver can read the message.

1.2. Message Authentication. The message should be authentic, and only the privileged nodes should be able to send the message. Furthermore, no eavesdropper should be able to masquerade the receiver by sending fake messages.

1.3. Integrity. The message received at the receiver side must exactly be the same as the sender sent.

1.4. Nonrepudiation. If there is any alteration in the message, whether due to the channel error or attacker's interference, the receiver must be able to recognize that and decline the message.

In [7–9], a few techniques have been discussed to address the communication security issues. In their proposed methods, a third party is involved to ensure the security of the communication network or setup, which depends upon the third-party network to communicate with remote end devices (REDs). Different encryption algorithms have been proposed in [10, 11] to secure the communication system using different cryptographic techniques, e.g., shift cipher and substitution cipher but, for a cryptanalyst, they are too simple to break or other encryption algorithms proposed in [12, 13]. A cryptanalyst can easily take the control of the system and can modify the control messages, as demonstrated in Figure 1. In Figure 1, an eavesdropper receives the message from the SCADA unit and modifies the messages and control command and sends that to the remote end device (RED) pretending to be the SCADA unit and hacks the system. In this way, control information becomes prone to the eavesdropper and loses authenticity and security. Specifically, in a smart-grid network, secure communication between the energy meters and the SCADA system requires a low cost and a secure communication setup with improved power efficiency. To provide remote access, a Raspberry Pi could also have been used for gateway purposes, just like [14, 15, 16] but, that consumes 3-4 times more power than this tiny ESP32. In this paper, we have proposed and implemented a secure and authentic communication system using an Advanced Encryption Standard (AES) algorithm. It is usually used for extreme confidential communication purposes, for military applications which do not involve any third party.

In Section 2, the implementation of different encryption algorithms on Arduino DRF1276G/ESP32 with LoRa for communication security have been discussed and compared. The comparison criteria are based upon their security and resistance against attacks. AES algorithm implementation steps and security have been explained in Section 3. Section 4 explains the local data logging on the SD card and the testing of the data transmission rate for different spreading factors. It also visualizes the comparison of received and sent messages at different spreading factors (at SF-7 and SF-12). In Section 5, two different gateways have been configured to upload the data to the server for remote access, and their pros and cons have been discussed briefly. Range testing and improvements in transmission range, using a mesh network algorithm, have been explained in Section 6. In Section 7, the results of AES and MAC implementation on Arduino DRF1276G LoRa and ESP32-LoRa have been shown and discussed.

2. Cryptographic Algorithms on Arduino with DRF1276G LoRa Module

To achieve the previously discussed four features of a secure and authentic communication system for microgrid, multiple encryption algorithms were implemented on the Arduino DRF1276G LoRa module but all cryptographic algorithms do not provide the equal secrecy level. Figure 2 shows a photo of the Arduino LoRa module used to implement the ciphers discussed below with their security against attacks.

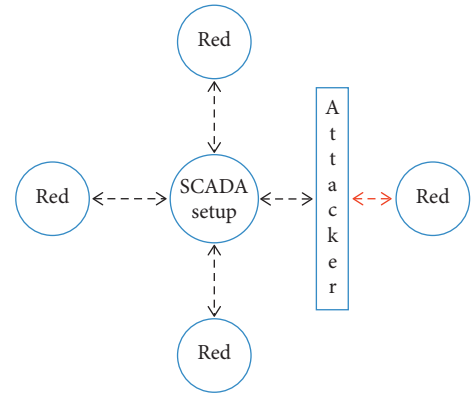


Figure 1: Eavesdropper masquerading the SCADA network.

2.1. Shift Cipher. In shift cipher, all characters of the message are shifted by the same number. For example, if the message is "abcdef" and the shift is by 3 characters, then after shift, "a" will go to "d," "b" to "e," and so on, as demonstrated below.

Plaintext	a	b	c	d	e	f
	↓	↓	↓	↓	↓	↓
Ciphertext	d	e	f	g	h	i

As there are only 25 possible shifts, it means that its key set has only 25 elements, and cipher can easily be decrypted within 25 attempts [17].

2.2. Affine Cipher. Key set of affine cipher is little bit larger than substitution cipher. In this technique, the ciphertext is calculated by solving a simple linear equation under modulo 26 because there are only 26 alphabets.

For instance, "y" indicates ciphertext and "x" indicates plaintext, and then $y = a * x + b$, where "a" and "b" are constants but less than 26. Its keyspace contains possible values for "b" and the possible values for which are 26 and 12, respectively. So, this cipher could be decrypted within $26 \times 12 = 312$ attempts [17].

2.3. Substitution Cipher. Substitution cipher gives much better security than the shift and affine cipher due to large key size. In the implementation, it is quite similar to the shift cipher but each plaintext character is not shifted by the same number, e.g.,

Plaintext	a	b	c	d	e	f
Ciphertext	d	z	h	k	a	f

The first character could be substituted by any of the other 25 characters, second character by any of the rest of 24 characters, and so on. In this way, possible key size becomes

$$|K| = 25 \times 24 \times 23 \ldots\ldots 1 = 25! \quad (1)$$

2.4. Transposition Cipher. In this cipher, the characters are not substituted, rather they are shuffled with each other within the plaintext block, e.g.,

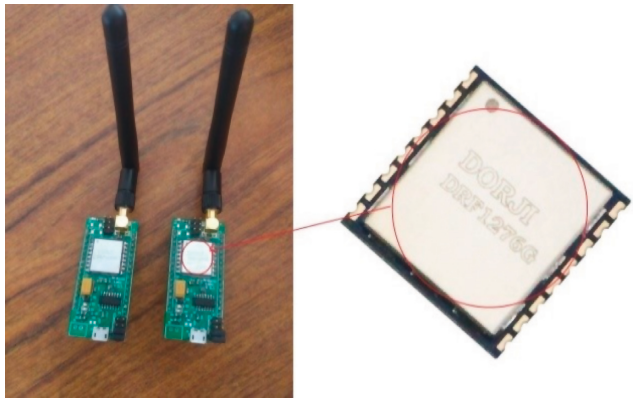


Figure 2: Arduino DRF1276G with LoRa module used for implementing cryptographic algorithms.

Its security depends upon the block size. If a block has "n" characters, then the key set will have total $n!$ possible values [17].

2.5. Hill Cipher. Hill cipher is based upon simple linear algebra, and its feature is that it is not an injective cipher. It is similar to the affine cipher, and the only difference is that it works on matrixes and columns of plain/ciphertext rather than characters. In this cipher, we assign numbers to all alphabet characters, e.g., $a = 0$, $b = 1$ and similarly $y = 24$ and $z = 25$, and use nxn square matrix as a key matrix to get the column matrix of ciphertext from the column vector of plaintext. For example, if the key matrix is $\begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix}$ and $\begin{bmatrix} D \\ R \end{bmatrix} = \begin{bmatrix} 4 \\ 18 \end{bmatrix}$ is supposed to be encrypted, then our cipher will be $\begin{bmatrix} 6 \\ 0 \end{bmatrix} = \begin{bmatrix} G \\ A \end{bmatrix}$, as shown in the following equation:

$$\begin{matrix} D \longrightarrow \\ R \longrightarrow \end{matrix} \begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 18 \end{bmatrix} = \begin{bmatrix} 58 \\ 26 \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \end{bmatrix} (\text{mod } 26). \quad (2)$$

However, hill cipher keyspace is m^{n^2} where "m" is the modulo and "n" is the size of the matrix, despite that it is vulnerable to chosen plaintext attack [17].

Although all previously discussed cryptographic techniques could be implemented for simple communication purpose, it could not be used for the SCADA system because of their vulnerability to attack by any cryptanalyst and the limited number of keyspace. That is why, we have implemented the AES algorithm to ensure the security of the SCADA system which not only has the keyspace of 2^{128} possible keys but also is nonlinear in nature and is flexible to change the pattern of output by changing the number of cascaded encryption rounds. To implement that, we were not able to use Arduino with the DRF1276G LoRa module, due to its small flash size and CPU limitations. To implement AES, we have used ESP32 which has not only enough flash, better CPU, and low cost, but also has very little power consumption (3.5–5 mW).

3. Implementation of AES Algorithm Using ESP32 and LoRa Module

In this implementation, MAC has been added to authenticate the communication, and AES implemented to encrypt the message, which is the most secure communication algorithm to all known attacks until now [18]. Figure 3 shows the flow chart of this implementation. Before sending any message, first it is encrypted using the AES encryption algorithm, and then, a 64-bit unique MAC is generated from the plaintext. Finally, the ciphertext and the MAC are concatenated and sent. Similarly, on the receiver side, first the MAC and the ciphertext are separated, then the MAC is verified, and the ciphertext is decrypted to process further. Figure 4 shows the step-by-step 10 round AES encryption and decryption. In the AES implementation, after generating a binary value string from the plaintext, there are four major steps which are repeated for each round.

3.1. Adding Round Key. The XOR sum is calculated by taking bit-wise XOR of each plaintext bit with respective key bit.

3.2. Substitute Bytes. After calculating the XOR sum, each byte (the pair of HEX characters) is replaced with the respective Rijndael table (a standard table of 256 values) to increase the confusion.

3.3. Shift Rows. After substitution, all 16 bytes are distributed to construct a 4×4 square matrix. In the resultant matrix, first row remains unchanged while the rest of the three rows (2^{nd}, 3^{rd}, and 4^{th}) are rotated left by 1, 2, and 3 bytes, respectively.

3.4. Mix Column. At this stage, the matrix left multiplication is applied using a standard 4×4 matrix on the results of shift row operation.

This algorithm not only has a large key set (2^{128} possible keys) but also is secure from many cryptanalysis algorithms like differential cryptanalysis, integration, linear, multiset, and many others like these.

In the AES algorithm, for each round, a new key is derived from the previous round key and the ciphertext of the previous round. From the test results, explained in the next section, it could be seen that each round ciphertext is entirely different from all others which is due to the implementation of binary-level encryption. The confusion created at each round and the propagation of confusion from one round to the next round makes it more secure.

3.5. MAC Generation. After the successful implementation of AES in the countermode, a unique and fixed size (64 bit) MAC is generated using the plaintext of the message. Its implementation ensures the authenticity of the message, and the receiver becomes able to verify whether the message has been modified by any eavesdropper, channel error, or not.

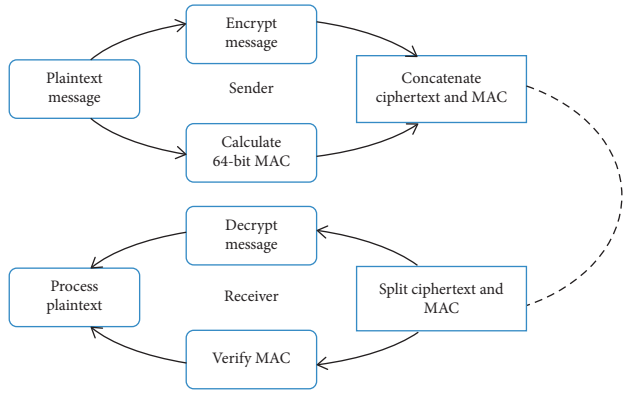


Figure 3: Flow chart of the implemented communication process.

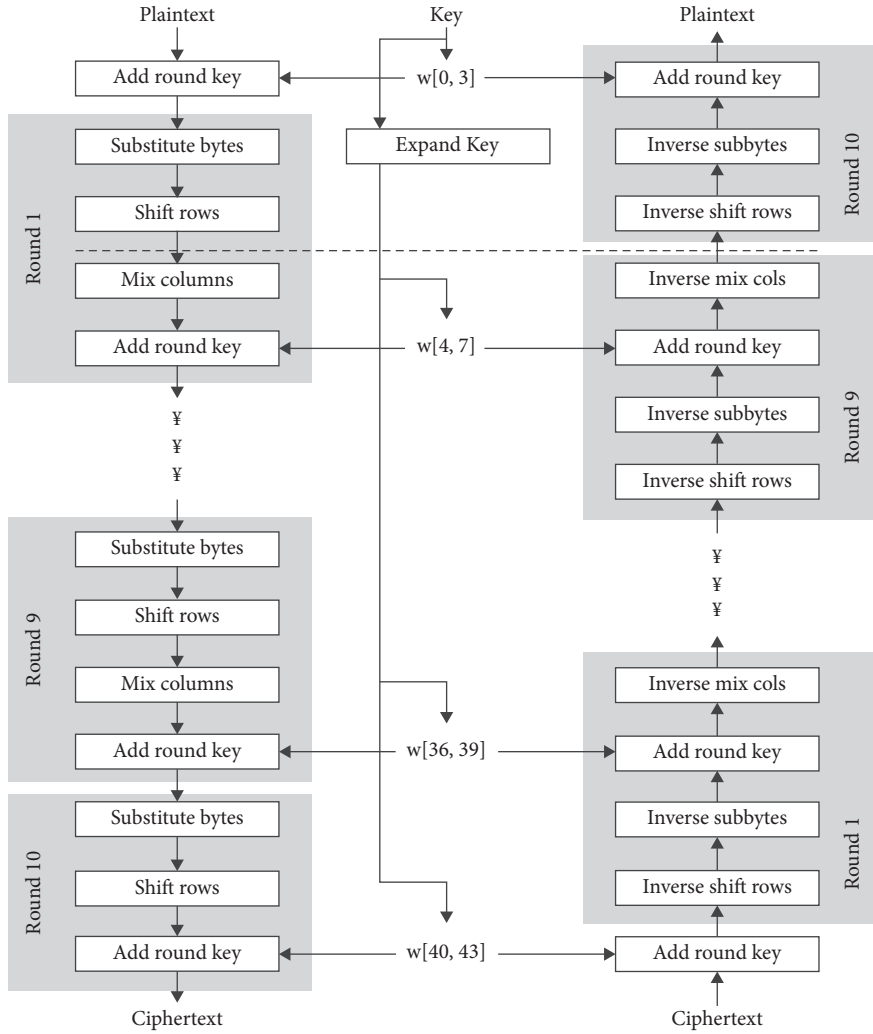


Figure 4: Step-by-step 10 round AES encryption and decryption [17].

4. Data Logging and Data Rate

Data logging is another feature added to this system in order to have a self-data backup mechanism and to avoid data loss due to any accidental failure in the communication system. The data which are supposed to be coming from local inverter/wind turbine is time-stamped and are stored after applying the AES encryption. Subsequently, the received data after extracting sender/receiver identity passwords and MAC are decrypted and verified to ensure that the received message is authentic. Finally, the data are time-stamped and stored in a separate received data file in CSV format. For the

local data storage, the configured SD card console is shown in Figure 5, and the logged data depend upon storage size and received data rate.

Although according to [19], the data rate of LoRa is around 27 kb/second which may go up to 50 kb/second, and the actual data rate depends upon the spreading factor. LoRa has six different spreading factors from SF7 to SF12 and is defined as $SF = \log 2\ (R_c/R_s)$ with R_c indicating chip rate and R_s indicating symbol rate [9, 19]. Data rate and range of the LoRa are SF dependent. High SF gives a better range but with a lower data rate. High SF also increases the probability of data loss, loss of authenticity, and the loss of integrity as well. On the other hand, a lower SF gives a better data rate but it does not support for long distances. With SF-7, a data rate of 27 kb/second could be achieved while with SF-12, one can transmit data up to 15 km with a poor data rate of a few hundred bits per second [20].

When an encryption algorithm is implemented, it also causes latency in processing [21]. It was observed during the data rate testing and the implementation of the AES encryption. In Figures 6(a) and 6(b), results show that overall time lapsed for encrypting and sending a message increases with increasing SF. Here, the message size was the same for all SFs, the encryption algorithm (AES) was also the same, and the increasing time lapsed was only different due to different spreading factors. From the graph, it could also be inferred that SF-12 takes approximately three times more as compared to SF-7. The relation between time elapsed and SF was also similar at the receiver end which corroborated this relation of SF and time elapsed.

Further visualization of this comparison was made by configuring one ESP32-LoRa with SF-7 and the other ESP32-LoRa with SF-12. In Figure 6(b), it can be seen that, in the lower half of the picture when sender and receiver were both configured with SF-7, the number of messages received was equal to the messages sent. While in the upper half of the picture, the sender was configured with SF-12 due to which a significant difference appeared between the rate of message sending and receiving (Figure 7).

```
SD Card Size: 7580MB
Listing directory: /
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
  DIR : /my_new_directory_1
Creating Dir: /mydir
Dir created
Listing directory: /
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
  DIR : /my_new_directory_1
  DIR : /mydir
Removing Dir: /mydir
Dir removed
Listing directory: /
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
  DIR : /my_new_directory_1
Listing directory: /my_new_directory_1
Writing file: /bytes.txt
File written
Appending to file: /bytes.txt
Message appended
1048576 bytes read for 2905 ms
1048576 bytes written for 4913 ms
Total space: 7563MB
```

Figure 5: Configuring SD card for data logging.

5. ESP32 versus Dragino Gateways and Alarming

The collected data were uploaded to a server for analysis and storage. To upload the data, two different gateways based upon ESP32 and dragino were tried but both had certain limitations. The configuration of an ESP32-based gateway is relatively difficult because it is to be configured as a gateway through coding while a dedicated dragino gateway is already available in the market with complete configuration and is more user-friendly. On the other hand, an ESP32-based gateway is much more cost-effective and power efficient. It hardly consumes 230–300 mW of power [22], while the dragino gateway takes around 12 W, and it requires 12 V DC for proper functioning while an ESP32 requires only 2.7–3.3 V DC. Overall, a dragino-based gateway consumes about 30 times more power than that of ESP32. Dragino compensates for this excess power consumption in terms of many other features. For example, it can serve up to 8 nodes simultaneously by communicating with each node at a different frequency [23], while ESP32 can support only 3 such nodes simultaneously.

In Figure 8(a), the configuration of an ESP32 based gateway is shown, and in Figure 8(b), The Things Network (TTN) data file is shown, in which data are being uploaded and could be accessed remotely. Figures 9(a) and 9(b) show the dragino controller with LoRa-based gateway configuration and its profile with the real-time data load.

6. Range Testing and Implementation of Mesh Network

The LoRa range was tested deploying one ESP32-LoRa at Memorial University and taking other EP32-LoRas to the Signal Hill, as shown in Figure 10. This setup supported a noise- and error-free communication for the distance of 3.85 km. Although its range is usually obstacle dependent and during another testing, it was observed that if the transmitter is at ground floor in the house window and a receiver is taken outside in neighboring streets, then the communication range drastically goes down, and they can communicate only up to the distance of 500–700 m. To address this issue, a network based upon mesh-like topology was implemented which gave better results.

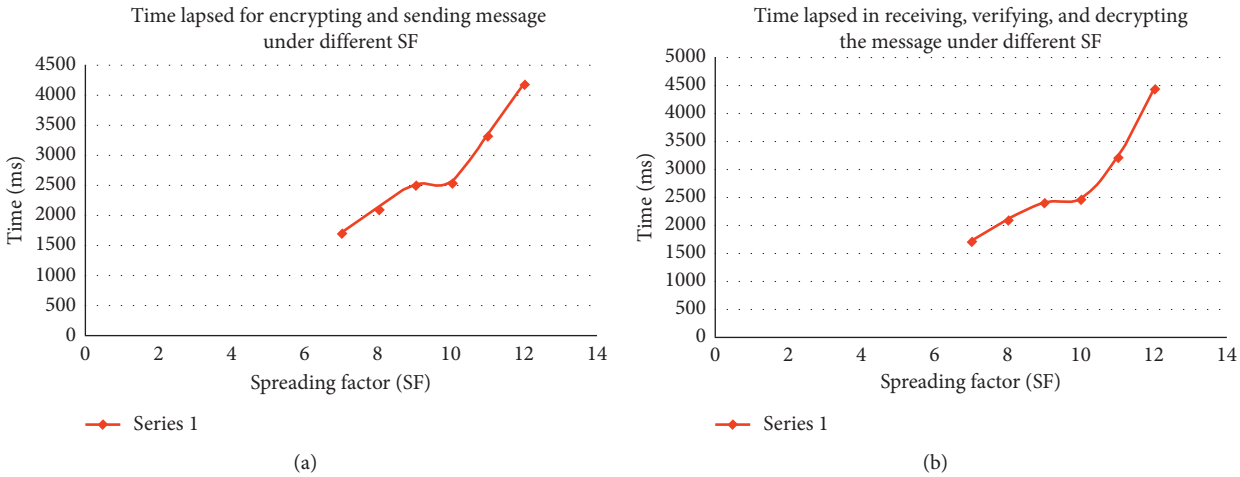


Figure 6: Latency for different SF and AES on the (a) sender end and (b) receiver end.

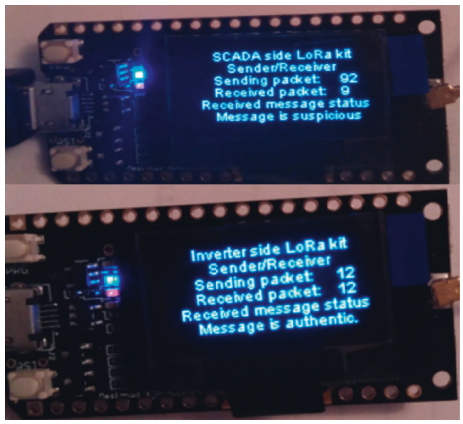
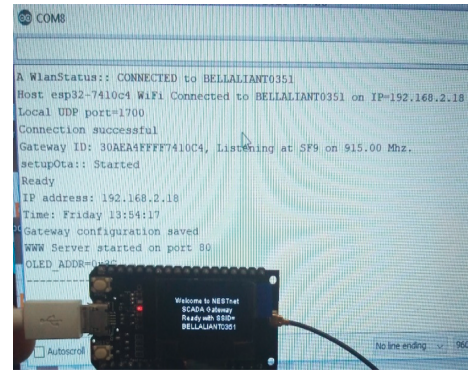
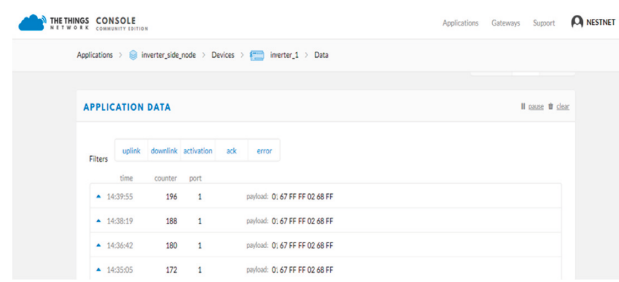


Figure 7: Configuring ESP32-LoRa sender and receiver at different SF.



(a)



(b)

Figure 8: (a) Configuring ESP32 as a gateway. (b) Uploading data on The Things Network.

To implement a mesh topology, each ESP32-Lora module of the network was assigned a unique identity code and was also fed with the directory of all other units' identity code directory. Before sending a message but after encryption and the MAC addition, the sender adds the target node identity code and its own identity code in the message string. The processing steps done at the receiver side are shown in the flow chart of Figure 11(a). An idle node goes into sleep mode for power saving, and whenever there is a message, it receives and parses the message packet into three parts:

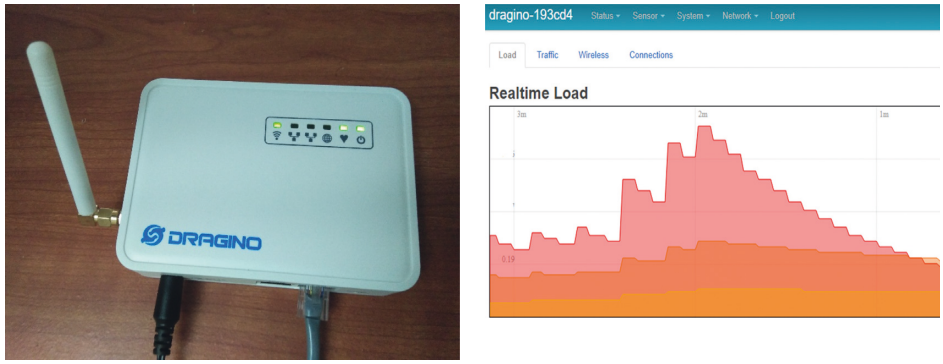
(a) Sender identity code

(b) Targeted receiver identity code

(c) Message packet with data and MAC information

The ID of the node is compared with the targeted receiver ID. If they are equal, this means that the node is the targeted receiver, and then, the received message packet is further parsed into encrypted message and MAC. After parsing the received message, the previously discussed message verification algorithm is applied, and after proving message authenticity, it is decrypted and then executed further. If the receiver ID is not equal to the targeted receiver ID, then the message is again packed in a single string as was received and is forwarded to the other nodes lying in the range.

In this way, if a message is sent from the central control unit for a node which does not lie in the range of that unit, then a node in the vicinity of the sender will receive that message and will forward to the next nearby node. A complete system flow chart is shown in Figure 11(b). In this figure, a node of level 2 lies out of the range of the central



(a) (b)

Figure 9: (a) A dedicated dragino LoRa gateway. (b) Dragino real time data load.

node, and a node of level 1 acts as a bridge for two-way communication between the control node and level-2 nodes.

After implementing mesh topology, its range was tested for two levels of nodes, and a significant improvement was observed. The nodes whose range was limited to 500–700 m achieved another 500 m in their coverage area, and its results can be seen in Figure 11(c). In this way, the range which was only obstacle dependent became the function of the number of levels between the sender and the receiver as well.

A complete system flow chart is shown in Figure 12. The data string (V, I, P, Q, and system health) is collected through sensors from distributed energy sources (e.g., wind turbine and solar panel) and their associated inverters. The string is then serially fed to the ESP32 unit. ESP32 encrypts the data and generates a unique MAC address for each message and adds to the encrypted message string. A unique identification code of the sender node and targeted receiver node is also added into that string before transmitting. A mesh node receives the message if the target node is not in range and forwards to give better coverage. Finally, when a message is received at the SCADA unit (targeted node), it verifies the sender-receiver ID, then parses the packet and confirms the message authenticity by verifying its MAC and then decrypts and uploads to the server through gateway under respective node ID.

Similarly, when a SCADA unit sends a command, it will first encrypt the message and will generate and add the MAC address into an encrypted message string and will also add the sender/receiver ID before sending. The receiver will follow the same steps; it will first match its own ID with the targeted node ID encapsulated in a message and then will verify the MAC and will decrypt before execution. If the message is not concerned with the receiver node, it will act as a bridge between the sender and the receiver and will transmit the message forward to improve coverage.

7. Results

All encryption algorithms discussed in Section 2 were tried, and finally, the combination of the AES algorithm implemented on ESP32 with LoRa for the SCADA system was chosen after comparing their security, authenticity for data, flexibility to change the key, and power consumption of the controllers. After selecting the AES algorithm, different controllers were tried and checked their compatibility with AES. Figure 13 shows the ESP32 with the LoRa module which costs about C$40 per set, consumes power around 5 mW, and supports the implementation of AES algorithm and AES with MAC as well.

Figure 14(a) shows the results of AES implementation on Arduino DRF1276G with the LoRa module. The results show that this controller cannot support even a single round of AES implementation due to small flash size and many other limitations. Finally, the ESP32 board was selected for this project due to its sufficient flash size and minimum power consumption.

The results of AES implementation are shown in Figure 14(b), in which a nine-round AES has been implemented on ESP32 with a LoRa module. It could be seen that the encrypted string is entirely different than the plaintext string which is due to bit-level changes made in the string during encryption. Furthermore, implementation of different numbers of encryption rounds generates a unique encrypted string which gives an additional advantage by increasing complexity. A different ciphertext for the same message can be generated by changing either number of rounds or changing key. Flexibility in changing key was achieved by externally connected buttons by either changing the number of encryption rounds or the key.

Figure 14(c) shows the results of the implementation of AES with MAC on ESP32 with the LoRa module, where the 192-bit received message is split into the 128-bit ciphertext and the 64-bit MAC. Decryption is applied on the ciphertext, and the plaintext is extracted from it after applying "n" decryption rounds. From that plaintext again, $n+1$ round ciphertext is calculated. An XOR sum is calculated between alternate bits of $n+1$ round ciphertext and respective plaintext bits. To check the authenticity of the message, calculated MAC is compared with the received MAC at the bit level, and even a single bit change in the received message is also detected in this comparator.

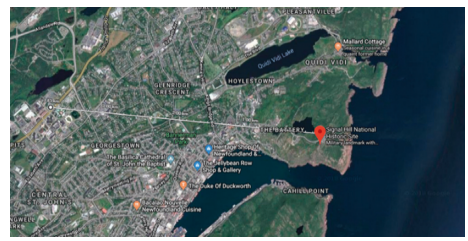


Figure 10: ESP32-LoRa range testing.

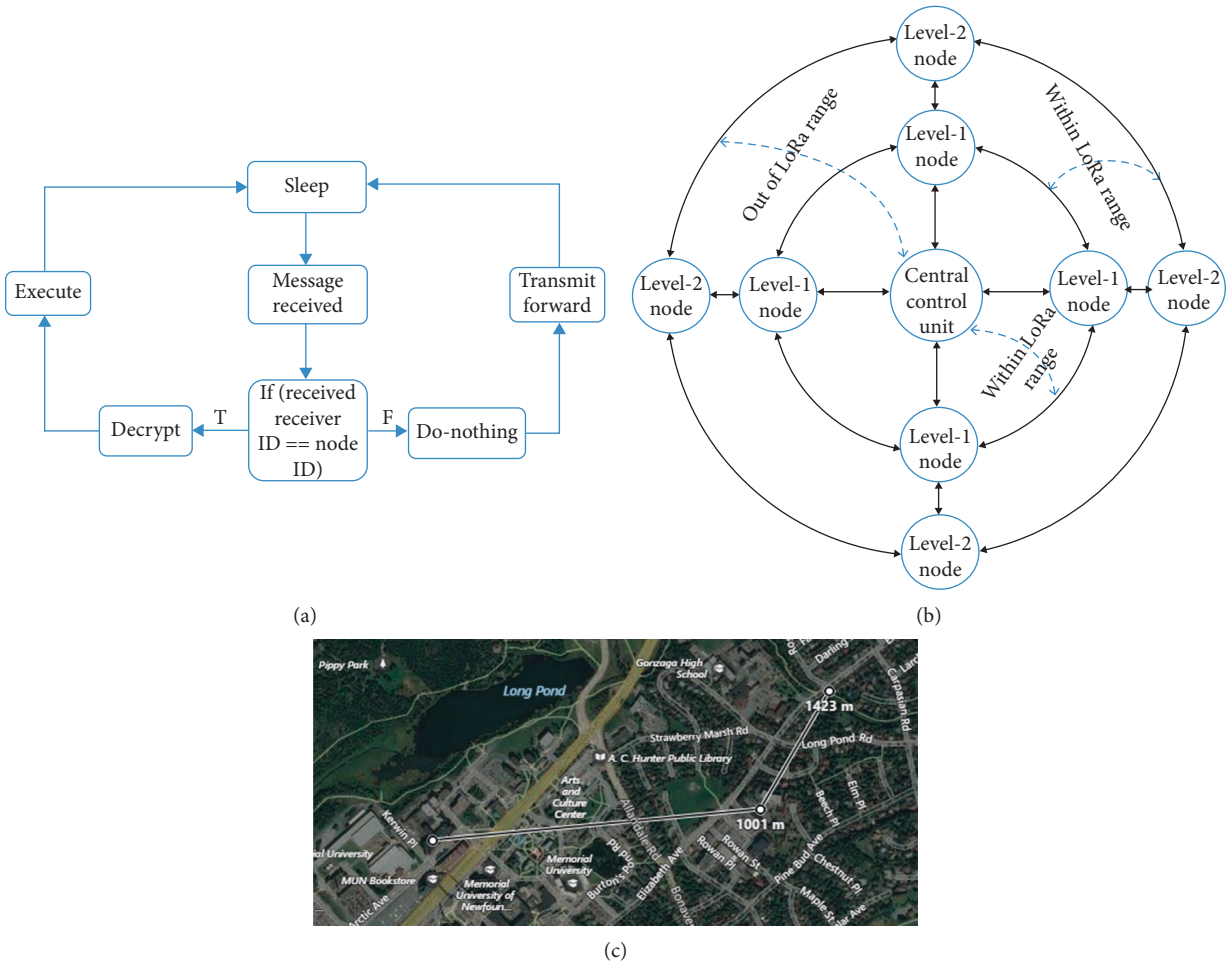


Figure 11: (a) Process flow chart for an intermediate level node. (b) Mesh network for improved LoRa range. (c) Range testing after implementing a mesh network.

Figures 14(d)–14(f) shows the results of one unit from every level of the mesh network. In Figure 14(d), the results show that the sender is "Node1" of level 2, the message is for the SCADA unit, and these addresses are followed by the encrypted message string and the MAC address. The receiver (SCADA) will calculate the MAC address from the encrypted string and will compare that with the MAC added after encrypted string to verify the message authenticity. In the results, the calculated MAC and received MAC are equal due to which verification status has been shown "authentic" and the message has been decrypted to process.

Figure 14(d) shows the results of a node which is acting as a bridge between the nodes of the level 2 and SCADA unit. It compares the targeted node ID with its device ID and finds that the message is for another node and sends the message forward without changing or processing. The results of Figure 14(f) show the results of a message sent from the SCADA unit, and the targeted node ID is "Node1." The string has the four parts such as sender ID, receiver ID, the encrypted message, and the MAC. The verification steps were followed before processing the message, and the final message status "authentic" proves the successful two-way communication of the nodes in the mesh network.

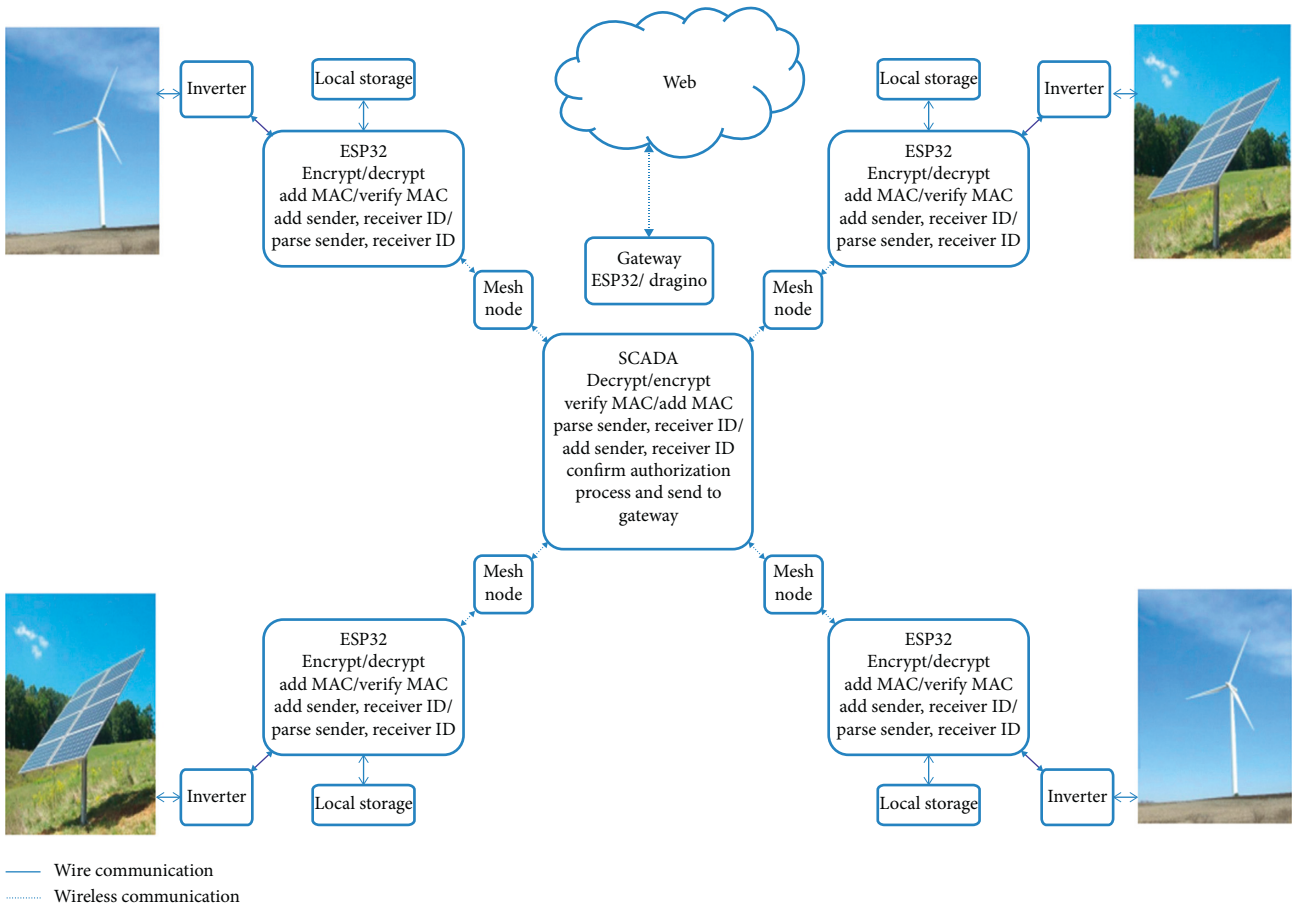


Figure 12: Complete system block diagram.

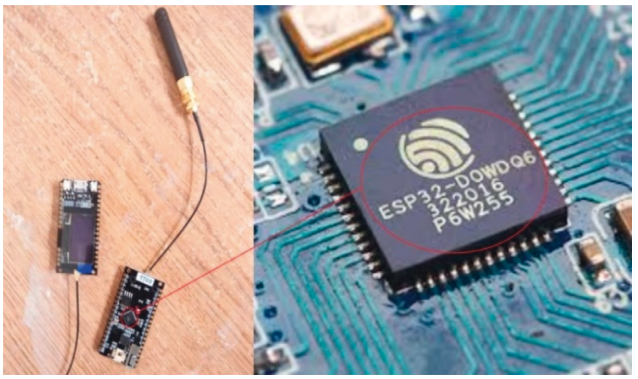


Figure 13: ESP32 with LoRa module used for AES implementation.

8. Conclusion

Implementation of AES cryptography with MAC for the SCADA system using ESP32 with LoRa was tested. It proved to be the best method of secured, authentic, and flexible communication. In a prototype system, a point-to-point secure and authentic communication has been achieved, for which the setup costs less than C$40 and consumes power less than 5 mW. The implemented encryption algorithm (AES) is also the most resistant cryptographic algorithm. It is only vulnerable to brute force attack which requires 2^{128} different keys to be tested to ensure the successful decryption. Moreover, its range was improved by implementing it in a mesh network. By implementing this network, as implemented here on a prototype network, a secure and sophisticated low-cost, remotely accessible, local, and remote data logging with broader coverage area using mesh network can be achieved. It seems to be the best solution to implement the SCADA system for the distributed operating units and integrated IoT network to achieve a secure and authentic communication system. In the distributed power generation and microgrids, its implementation for the

```
pre-encrypt(plain_text):        0123456789ABCDEF0123456789ABCDEF
Ciphet-text after 1
Ciphet-text after 2 rounds:
Ciphet-text after 3 rounds:
Ciphet-text after 4 rounds:
Ciphet-text after 5 rounds:
Ciphet-text after 6 rounds:
Ciphet-text after 7 rounds:
Ciphet-text after 8 rounds:
Ciphet-text after 9 rounds:
final ciphertext:
```

(a)

```
pre-encrypt(plain_text):        0123456789ABCDEF0123456789ABCDEF
Ciphet-text after 1 rounds:     4023CABB284333AC463817F42893333D
Ciphet-text after 2 rounds:     4716512657507AE6F0BFB407EBEC5817
Ciphet-text after 3 rounds:     89C5F20861099F9EEB73639A0BE8D3EA
Ciphet-text after 4 rounds:     F1D452604C3119F03577B2790FF6A34D
Ciphet-text after 5 rounds:     92A6C59992FDB6A8A8452D28E3764214
Ciphet-text after 6 rounds:     A0DAD27C43FE5684D8349F6EFA113858
Ciphet-text after 7 rounds:     D71F17D9383AFE1FF08EA6657040EED5
Ciphet-text after 8 rounds:     512E95839F96762F9C544D00A66FFA17
Ciphet-text after 9 rounds:     FF75BAA2661F246560821DBD47D73AB2
final ciphertext:               FF75BAA2661F246560821DBD47D73AB2
```

(b)

```
pre-encrypt(plain_text):        0123456789ABCDEF0123456789ABCDEF
Ciphertext without MAC is:      F9F026E7CD825F05A559FE74E4656FE9
Decrypted Plaintext:            0123456789ABCDEF0123456789ABCDEF
ciphertext with MAC:            F9F026E7CD825F05A559FE74E4656FE9410DFC5C95DFF8CE
Received_MAC:                   410DFC5C95DFF8CE
Calculated_MAC:                 410DFC5C95DFF8CE
MAC verification status:        Message is authentic.
Verified decrypted message is: 0123456789ABCDEF0123456789ABCDEF
```

(c)

```
pre-encrypt(plain_text):        0123456789ABCDEF0123456789ABCDEF
ciphertext with MAC:            F9F026E7CD825F05A559FE74E4656FE9410DFC5C95DFF8CE
Sending message:                Node15CADAF9F026E7CD825F05A559FE74E4656FE9410DFC5C95DFF8CE
ReceivedCiphertext with MAC is:F9F026E7CD825F05A559FE74E4656FE9410DFC5C95DFF8CE
Received_MAC:                   410DFC5C95DFF8CE
Calculated_MAC:                 410DFC5C95DFF8CE
MAC verification status:        Message is authentic.
Verified decrypted message is: 0123456789ABCDEF0123456789ABCDEF
with RSSI: -37
```

Sender ID, Receiver ID, Encrypted message, MAC

(d)

```
pre-encrypt(plain_text):        0123456789ABCDEF0123456789ABCDEF
Sending message:                F9F026E7CD825F05A559FE74E4656FE9410DFC5C95DFF8CE
Sending message:                Meshlhijk1F9F026E7CD825F05A559FE74E4656FE9410DFC5C95DFF8CE
Message is not concerned to me
Message forwarded
```

(e)

Figure 14: Continued.

```
SCADA side LoRa Sender/Receiver
LoRa setup configured successfully!
pre-encrypt(plain_text):       0123456789ABCDEF0123456789ABCDEF
ciphertext with MAC:           F9F026E7CD825F05A559FE74E4656FE9410DFC5C95DFF8CE
Sending message:               SCADANode1F9F026E7CD825F05A559FE74E4656FE9410DFC5C95DFF8CE
ReceivedCiphertext with MAC is:F9F026E7CD825F05A559FE74E4656FE9410DFC5C95DFF8CE
Received_MAC:                  410DFC5C95DFF8CE
Calculated_MAC:                410DFC5C95DFF8CE
MAC verification status:       Message is authentic.
Verified decrypted message is: 0123456789ABCDEF0123456789ABCDEF
with RSSI: -26
```

(f)

Figure 14: (a) AES implementation on Arduino DRF1276G with LoRa module. (b) AES implementation results on ESP32 with LoRa module. (c) Implementation of AES with MAC on ESP32 with LoRa module. (d) Results of terminal node in a mesh network. (e) Intermediate level node forwarding message back and through in the mesh network. (f) Message sent from SCADA unit with the targeted node ID.

SCADA system and protects power generation from the hostile actors, and it can send wireless data over many kilometers with low-cost and negligible power consumption.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

This work is the extension of previous conference paper "Low-Cost and Secure Communication System for Remote Microgrids using AES Cryptography on ESP32 with LoRa Module" presented at EPEC 2018. Here, it has been significantly extended and elaborated.

Conflicts of Interest

The authors declare that all used software devices were selected on professional basis. Furthermore, the authors certify that there are no actual or potential conflicts of interest in relation to this article.

Acknowledgments

This research was funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada. The authors would like to acknowledge the financial support of NSERC and technical support of friends, family, and the Memorial University of Newfoundland.

References

[1] C. Mavrokefalidis, D. Ampeliotis, and K. Berberidis, "A study of the communication needs in micro-grid systems," in *Proceedings of the General Assembly and Scientific Symposium of the International Union of Radio Science (URSI GASS) 2017*, pp. 1–4, Montreal, Canada, August 2017.

[2] A. García-Domínguez, "Enabling SCADA cluster and cloud for smart grid using hierarchical multicast; the PTMF framework," in *Proceedings of the IEEE International Conference on Industrial Technology*, vol. 2015, pp. 218–225, Seville, Spain, June 2015.

[3] H. H. Safa, D. M. Souran, M. Ghasempour, and A. Khazaee, "Cyber security of smart grid and SCADA systems, threats and risks," in *Proceedings of the CIRED Workshop 2016*, pp. 1–4, Helsinki, Finland, June 2016.

[4] E. Bou-Harb, C. Fachkha, M. Pourzandi, M. Debbabi, and C. Assi, "Communication security for smart grid distribution networks," *IEEE Communications Magazine*, vol. 51, no. 1, pp. 42–49, 2013.

[5] A. Tanenbaum, "Network security," in *Computer Networks*, pp. 767–790, Pearson, London, UK, 5th edition, 2011.

[6] H. Su, M. Qiu, and H. Wang, "Secure wireless communication system for smart grid with rechargeable electric vehicles," *IEEE Communications Magazine*, vol. 50, no. 8, pp. 62–68, 2012.

[7] D. NamdeoHire, "Secured wireless data communication," *International Journal of Computer Applications*, vol. 54, no. 1, pp. 27–30, 2012.

[8] A. A. P. Ratna and R. F. Sari, "A test bed implementation of secure and lightweight privacy preservation mechanism using scrambled Fibonacci and XOR for ZigBee," in *Proceedings of the Region 10 Conference, TENCON 2017*, pp. 863–868, George, Malaysia, November 2017.

[9] Y.-S. Tsai, C.-Y. Chu, M.-C. Li, Y.-H. Lin, and P. Chen, "Intelligent DC power monitoring system and sensor network based on ZigBee-equipped smart sockets," in *Proceedings of the 5th International Symposium on Next-Generation Electronics, ISNE 2016*, Hsinchu, Taiwan, May 2016.

[10] A. Shahzad, Y. G. Kim, and A. Elgamoudi, "Secure IoT platform for industrial control systems," in *Proceedings of the 2017 International Conference on Platform Technology and Service*, Busan, Korea, February 2017.

[11] A. V. D. M. Kayem, H. Strauss, S. D. Wolthusen, and C. Meinel, "Key management for secure demand data communication in constrained micro-grids," in *Proceedings of the IEEE 30th International Conference on Advanced Information Networking and Applications Workshops*, pp. 585–590, Taipei, Taiwan, March 2016.

[12] J. L. Tsai and N. W. Lo, "Secure anonymous key distribution scheme for smart grid," *IEEE Transactions on Smart Grid*, vol. 7, p. 1, 2016, https://ieeexplore.ieee.org/abstract/document/7134810.

[13] X. Miao and X. Chen, "Cyber security infrastructure of smart grid communication system," in *Proceedings of the China*

International Conference on Electricity Distribution, pp. 5-6, Shanghai, China, September 2012.

[14] C.-S. Choi, J.-D. Jeong, I.-W. Lee, and W.-K. Park, "LoRa based renewable energy monitoring system with open IoT platform," in *Proceedings of the International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1-2, Honolulu, HI, USA, January 2018.

[15] H.-R. Lee, W.-J. Kim, K. Park, H.-J. Cho, and C.-H. Lin, "Development of an easy payment system based on IoT gateway," in *Proceedings of the International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1–3, Honolulu, HI, USA, January 2018.

[16] R. G. Anvekar, R. M. Banakar, and R. R. Bhat, "Design alternatives for end user communication in IoT based system model," in *Proceedings of the IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*, Chennai, India, 2017, https://ieeexplore.ieee.org/document/8273698.

[17] W. Stallings, *"Cryptography and Network Security"*, Pearson, London UK, 5th edition, 2011.

[18] P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, "A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and blowfish," *Procedia Computer Science*, vol. 78, pp. 617–624, 2016.

[19] T. W. F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, and J. Melia-Segui, "Understanding the limits of LoRaWAN," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[20] B. Jalaian, T. Gregory, N. Suri, S. Russell, L. Sadler, and M. Lee, "Evaluating LoRaWAN-based IoT devices for the tactical military environment," in *Proceedings of the IEEE World Forum on Internet of Things, WF-IoT 2018*, Limerick, Ireland, May 2018.

[21] S. J. Habib, M. Ahmad, M. A. Syed Hassan Ahmed, and J. J. P. C. Rodrigues, "Speeding up the internet of Things," *IEEE Consumer Electronics Magazine*, vol. 7, no. 6, pp. 31–37, 2018.

[22] L. Salman, S. Salman, and S. Jahangirian, "Energy efficient IoT-based smart home," in *Proceedings of the 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Reston, VA, USA, December 2016.

[23] Dragino, *LG01 LoRa Gateway User Manual*, Queensland University of Technology, Brisbane, Queensland, 2018.