Sensors and Materials, Vol. 34, No. 6 (2022) 2189–2197
MYU Tokyo

S & M 2965

Information Security in Wireless Water Flow and Leakage Alarm System

Chih-Chun Chang, Yao-Yu Lee, Ting-Yu Hou, and Chin-Chung Yu*

Department of Applied Physics, National University of Kaohsiung,
Kaohsiung University Rd., Nanzih District, Kaohsiung 811, Taiwan, R.O.C.

(Received December 6, 2021; accepted April 20, 2022)

Keywords: ESP32, IoT, information security, water flow meter, leakage detection

Toward improving water pipe leakage detection, we use an ESP32 board, sensors, and Internet of Things (IoT) technology to achieve the flow rate detection and throttling of water. We use a Message Queuing Telemetry Transport server as a broker to receive and transmit the flow data and leakage alarm, respectively. The leakage detection and LINE notification are realized using a Python program. For the information security of IoT, we use Galois/Counter Mode block encryption for packaging IoT messages. The transmission speed of the encrypted message has no notable difference from that of the unencrypted one when the content exceeds 1000 bytes.

1. Introduction

Water is an indispensable resource for living creatures. It maintains the daily lives of human beings and is also an integral part of the global environment. According to Water Resources Agency statistics in Taiwan,(1,2) the average annual rainfall of Taiwan is as high as 2500 mm. However, rain cannot be effectively stored and utilized owing to its high intensity in Taiwan and Taiwan's uneven seasonal distribution and steep terrain. In the Environment Sustainable Index evaluation,(3) Taiwan is the 18th most water-scarce country in the world.

When Taiwan Water Corporation was established in 1963, the corporation adopted low-cost economic management to increase the penetration rate of tap water quickly.(4) Many water pipelines were old and in poor condition because of human and natural damage over the years. There was severe water leakage, resulting in the loss of a large amount of water. In recent years, global climate change has highlighted the importance of water resources. At present, the average water fee in Taiwan is only 10.11 NT dollars per 1000 liters,(5) which is relatively cheap compared with those in other developed countries. How to effectively use water resources is one of the environmental sustainability issues currently being focused on in Taiwan.

With continuous technological advances, governments of various countries are actively promoting the Internet of Things (IoT) industry through multiple methods and policies. The IoT has addressed multiple needs in numerous fields. The number of IoT devices used in 2021 alone was 48 billion units in the world.(6) The IoT has been applied in a wide range of locations

*Corresponding author: e-mail: yucc@nuk.edu.tw
https://doi.org/10.18494/SAM3811

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

including shopping malls, offices, factories, and private residences, and it is revolutionizing industry. However, it is necessary to address its information security problems.

Water companies often use a gauge to record water consumption for water leakage detection. However, a gauge cannot provide a real-time leakage alarm. Some researchers proposed the detection of water pressure and flow rate to realize real-time leakage detection.(7–9) In recent studies, a combination of microcontrollers, the IoT, and miniature sensors have been used to detect water leakage and notify users.(10,11) Moreover, Secure Sockets Layer (SSL)/Transport Layer Security (TLS) protocols are used in the encryption of IoT data.(12,13) It is necessary to develop a water leakage detection method based on the IoT with a secure mechanism.

In this study, we took the leakage detection of a water pipe as the theme and realized the throttling of water resources using IoT techniques. We used TLS encryption to increase data transmission safety and notified the user of water leakage via a real-time LINE notification.

2. Experimental Technologies

An ESP32 series development board manufactured by Espressif Information Technology was used as a microcontroller in this study. The ESP32 board can be programmed and burned by Arduino IDE. It has an additional CPU core and a large number of GPIO pins, and supports Wi-Fi, Bluetooth 4.2, and Bluetooth Low Energy at the same time. In this study, Hall sensors were equipped on the ESP32 board and used to detect the flow rate of water. The ESP32 board transmitted the data of the flow rate to a Message Queuing Telemetry Transport (MQTT) server. The leakage alarm was sent to the user via a LINE notification.

2.1 Water flow meter

Two water flow meters comprising a rotor with turbine blades were connected serially with the water pipeline, as shown in Figs. 1(a) and 2(a). A small magnet was placed firmly at the end of one of the blades of each flow meter. A Hall sensor near the blades outputted an electric pulse as the magnet passed by, and the output pulse was obtained by the ESP32 board with a built-in interrupt function as shown in Fig. 1(b). Then, we used the millis() function to obtain the cumulative number of pulse signals per second, known as pulse frequency, for the Hall sensor.

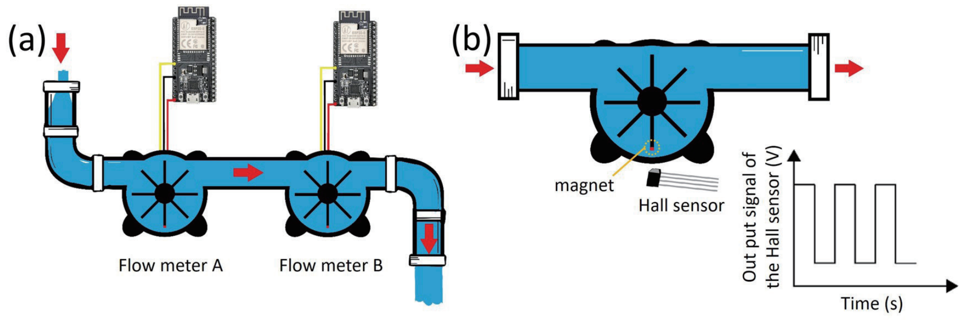


Fig. 1. (Color online) (a) Water flow meters based on Hall effect sensor. (b) Diagram of the experimental setup.

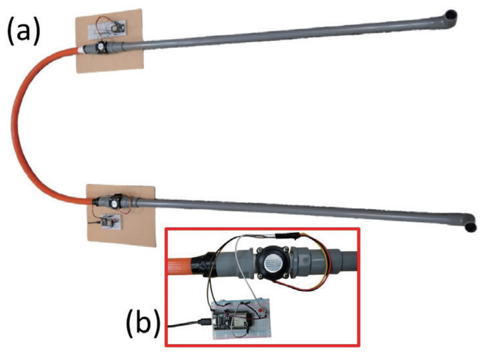


Fig. 2. (Color online) (a) Overall experimental setup and (b) closer view of a flow meter.

When the fluid flowed, the friction of the bearing was assumed to be negligible so that the flow rate was proportional to the rotational speed of the rotor and, consequently, the output pulse frequency of the Hall sensor.(14)

2.2 Calibration method

Water with a known feed rate flowed through the pipe, and the ESP32 board recorded the pulse frequency of the Hall sensor. Figure 3 shows the pulse frequencies under different water flow rates. Then, the following relationship between the flow rate and the pulse frequency was obtained by linear fitting:

$$\textit{Water flow rate}\ [\mathrm{L/min}] = \frac{\textit{Pulse frequency}\ [\mathrm{Hz}]}{7.1\ [\mathrm{Hz}\cdot\mathrm{min/L}]}. \tag{1}$$

For the detection of water leakage, at least two flow meters must be connected in series. The flow rates were recorded simultaneously by two ESP32 boards. When there is a drop in the value between the two flow meters, there is water leakage from the pipeline.

2.3 MQTT server

MQTT is a commonly used communication technology for the IoT and is based on Publish/Subscribe and the TCP/IP protocol. In this study, we used the MQTT communication protocol to transmit the monitoring data of the water flow to the server via a Wi-Fi connection. A broker is also needed to implement the heavy message transmission protocol. We installed Mosquitto open-source software as the broker on a PC and used MQTT certificate authentication as the data encryption for the IoT devices. After the ESP32 boards sent the flow rate messages to the MQTT server through the Wi-Fi connection, we used a Python program to determine whether there was water leakage, then a LINE notification was sent to the user's mobile phone.

In the MQTT message format, the header message, including the message type, transmission quality (QoS), and so forth, only occupies two bytes. The main message content can be up to

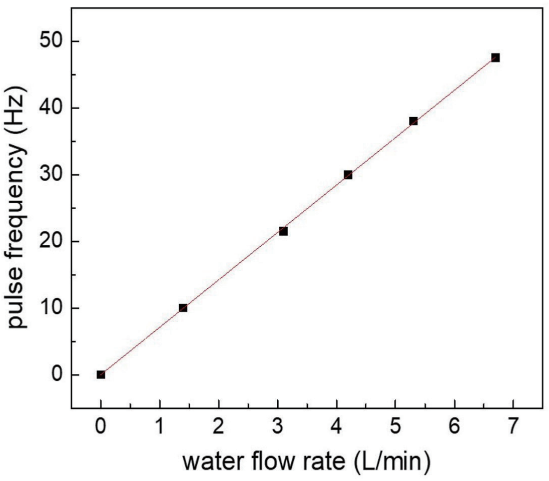


Fig. 3. (Color online) Plot of pulse frequency versus water flow rate.

256 MB. In addition, clients only need to subscribe or publish topics to the agent, and each client does not need to know the IP addresses of other clients.(15) Therefore, MQTT can achieve lightweight communication with a small packet transmission volume in data transmission. Figure 4 shows an example of the overall MQTT transmission process.

2.4 Message encryption

We used the SSL/TLS transmission security protocol to encrypt the message output from the ESP32 board based on the MQTT transmission protocol. We then transmitted the message entirely to the server and subsequent clients.(16)

3. Results and Discussion

3.1 Detection of water leakage

The Arduino ESP32 board calculated the water flow rate from the pulse signal generated from the Hall sensor and then sent the flow data through the MQTT communication protocol via the Wi-Fi connection. Then, the flow data were saved as a CSV file on the MQTT server, as shown in Fig. 5. Here, we subscribed to the MQTT message on the client side using a Python program. Then, the LINE Notify service was used to send push notifications to the user's mobile phone.

Two flow meters, A and B, were used for leakage detection, as shown in Figs. 1(a) and 2(a). We used a knife to slightly cut the plastic pipeline between the two flow meters to simulate water leakage. The flow rate of the two meters was monitored simultaneously. A flow rate of flow meter A higher than that of flow meter B indicates water leakage. When the leakage rate was higher than 0.1 L/min, the leakage alarm was triggered. A LINE notification was automatically sent to the users to inform them of the water leakage, as shown in Fig. 6.

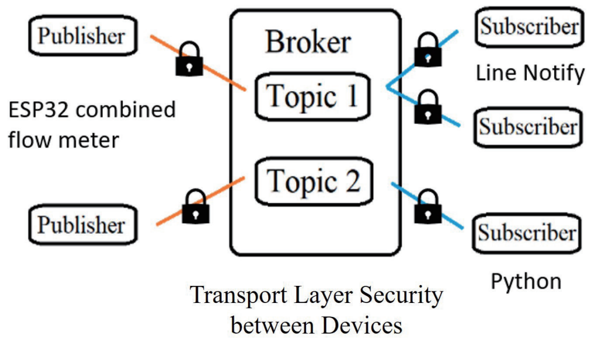
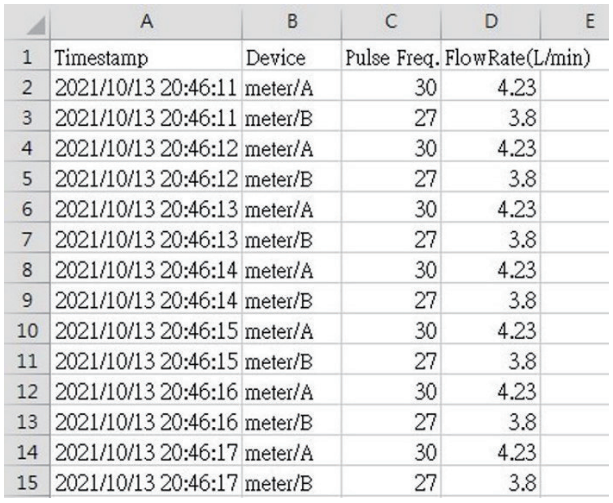
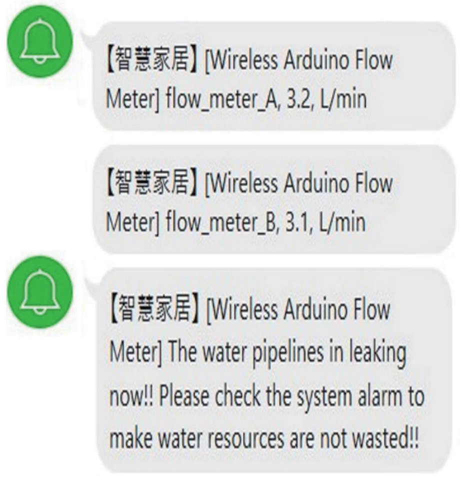


Fig. 4. (Color online) MQTT transmission process.



	A	B	C	D	E
1	Timestamp	Device	Pulse Freq.	FlowRate(L/min)	
2	2021/10/13 20:46:11	meter/A	30	4.23	
3	2021/10/13 20:46:11	meter/B	27	3.8	
4	2021/10/13 20:46:12	meter/A	30	4.23	
5	2021/10/13 20:46:12	meter/B	27	3.8	
6	2021/10/13 20:46:13	meter/A	30	4.23	
7	2021/10/13 20:46:13	meter/B	27	3.8	
8	2021/10/13 20:46:14	meter/A	30	4.23	
9	2021/10/13 20:46:14	meter/B	27	3.8	
10	2021/10/13 20:46:15	meter/A	30	4.23	
11	2021/10/13 20:46:15	meter/B	27	3.8	
12	2021/10/13 20:46:16	meter/A	30	4.23	
13	2021/10/13 20:46:16	meter/B	27	3.8	
14	2021/10/13 20:46:17	meter/A	30	4.23	
15	2021/10/13 20:46:17	meter/B	27	3.8	

Fig. 5. Recorded CSV data file.



【智慧家居】[Wireless Arduino Flow Meter] flow_meter_A, 3.2, L/min

【智慧家居】[Wireless Arduino Flow Meter] flow_meter_B, 3.1, L/min

【智慧家居】[Wireless Arduino Flow Meter] The water pipelines in leaking now!! Please check the system alarm to make water resources are not wasted!!

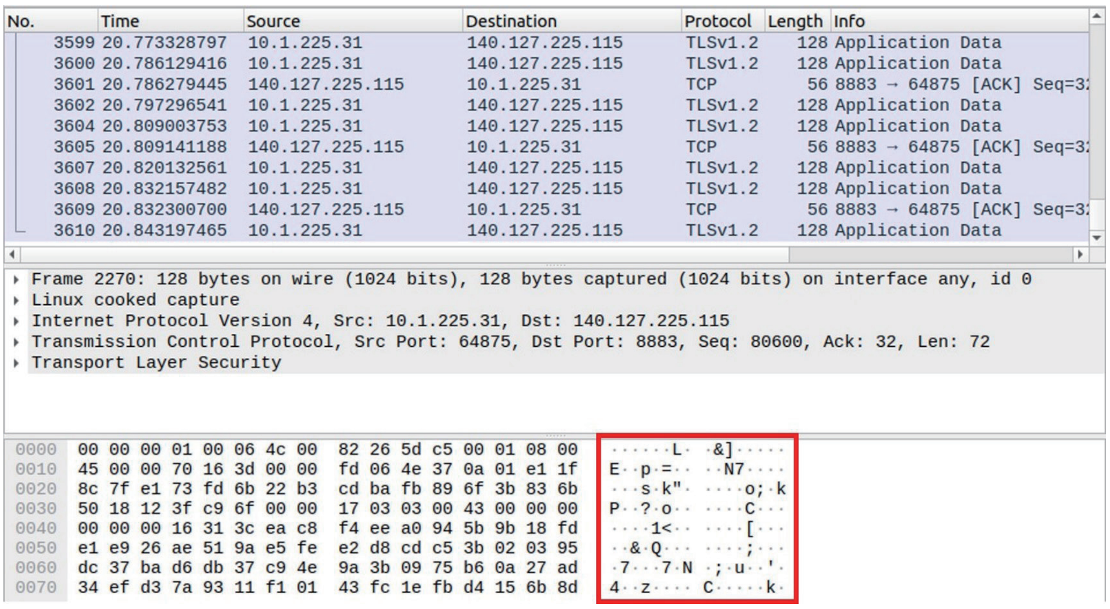
Fig. 6. (Color online) LINE Notify alarm messages.

3.2 Information security of IoT

Flow meter B used TLS encryption during data transmission with the transmission port set to 8883. The other transmitted messages in plain text without encryption with the transmission port set to 1883. Wireshark network packet monitoring software was installed on the MQTT broker server to transmit and capture the IoT data.

Wireshark successfully retrieved the encrypted and unencrypted MQTT messages, as shown in Figs. 7 and 8, respectively. Although one cannot solve the transmission data of the encrypted IoT data, the unencrypted message clearly showed the transmission topic and flow data. For example, according to Fig. 8, flow meter A sent out a message with a pulse frequency of 30 Hz and a flow rate of 4.23 L/min.

It is crucial to maintain a certain data transmission speed. To evaluate the transmission efficiency of the encrypted data, we forced the ESP32 client to continuously send messages to

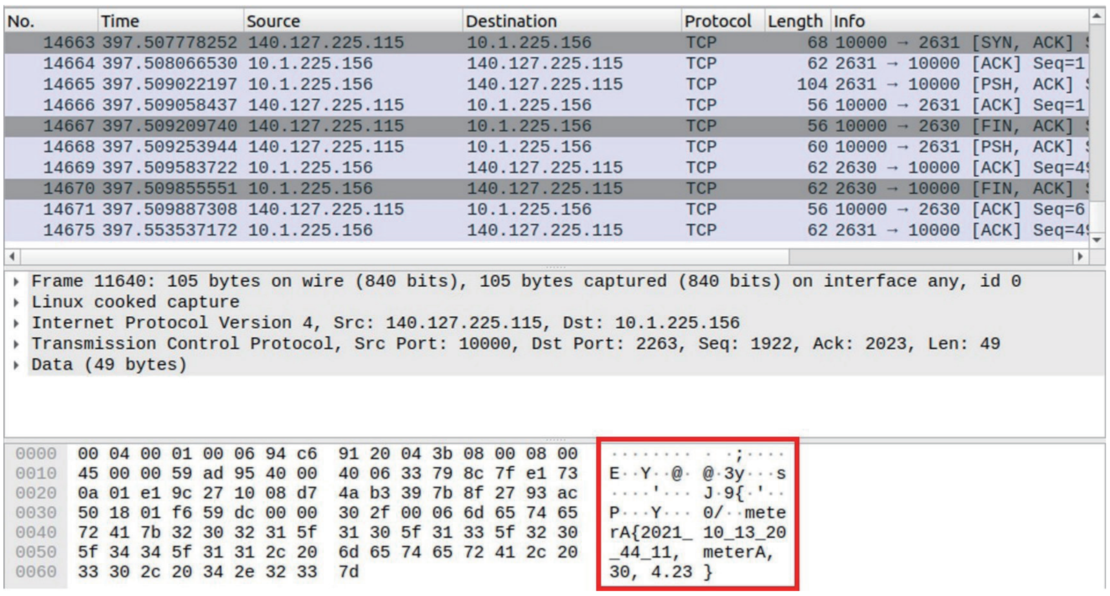


```
No.   Time          Source           Destination      Protocol Length Info
 3599 20.773328797  10.1.225.31      140.127.225.115  TLSv1.2   128 Application Data
 3600 20.786129416  10.1.225.31      140.127.225.115  TLSv1.2   128 Application Data
 3601 20.786279445  140.127.225.115  10.1.225.31      TCP        56 8883 → 64875 [ACK] Seq=3:
 3602 20.797296541  10.1.225.31      140.127.225.115  TLSv1.2   128 Application Data
 3604 20.809003753  10.1.225.31      140.127.225.115  TLSv1.2   128 Application Data
 3605 20.809141188  140.127.225.115  10.1.225.31      TCP        56 8883 → 64875 [ACK] Seq=3:
 3607 20.820132561  10.1.225.31      140.127.225.115  TLSv1.2   128 Application Data
 3608 20.832157482  10.1.225.31      140.127.225.115  TLSv1.2   128 Application Data
 3609 20.832300700  140.127.225.115  10.1.225.31      TCP        56 8883 → 64875 [ACK] Seq=3:
 3610 20.843197465  10.1.225.31      140.127.225.115  TLSv1.2   128 Application Data

▸ Frame 2270: 128 bytes on wire (1024 bits), 128 bytes captured (1024 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.1.225.31, Dst: 140.127.225.115
▸ Transmission Control Protocol, Src Port: 64875, Dst Port: 8883, Seq: 80600, Ack: 32, Len: 72
▸ Transport Layer Security

0000  00 00 00 01 00 06 4c 00  82 26 5d c5 00 01 08 00   ······L· ·&]····
0010  45 00 00 70 16 3d 00 00  fd 06 4e 37 0a 01 e1 1f   E··p·=·· ··N7····
0020  8c 7f e1 73 fd 6b 22 b3  cd ba fb 89 6f 3b 83 6b   ···s·k"· ····o;·k
0030  50 18 12 3f c9 6f 00 00  17 03 03 00 43 00 00 00   P··?·o·· ····C···
0040  00 00 00 16 31 3c ea c8  f4 ee a0 94 5b 9b 18 fd   ····1<·· ····[···
0050  e1 e9 26 ae 51 9a e5 fe  e2 d8 cd c5 3b 02 03 95   ··&·Q··· ····;···
0060  dc 37 ba d6 db 37 c9 4e  9a 3b 09 75 b6 0a 27 ad   ·7···7·N ·;·u··'·
0070  34 ef d3 7a 93 11 f1 01  43 fc 1e fb d4 15 6b 8d   4··z···· C·····k·
```

Fig. 7. (Color online) Encrypted message intercepted by Wireshark.



```
No.    Time           Source           Destination      Protocol Length Info
 14663 397.507778252  140.127.225.115  10.1.225.156     TCP        68 10000 → 2631 [SYN, ACK] S
 14664 397.508066530  10.1.225.156     140.127.225.115  TCP        62 2631 → 10000 [ACK] Seq=1
 14665 397.509022197  10.1.225.156     140.127.225.115  TCP       104 2631 → 10000 [PSH, ACK] S
 14666 397.509058437  140.127.225.115  10.1.225.156     TCP        56 10000 → 2631 [ACK] Seq=1
 14667 397.509209740  140.127.225.115  10.1.225.156     TCP        56 10000 → 2630 [FIN, ACK] S
 14668 397.509253944  140.127.225.115  10.1.225.156     TCP        60 10000 → 2631 [PSH, ACK] S
 14669 397.509583722  10.1.225.156     140.127.225.115  TCP        62 2630 → 10000 [ACK] Seq=4
 14670 397.509855551  10.1.225.156     140.127.225.115  TCP        62 2630 → 10000 [FIN, ACK] S
 14671 397.509887308  140.127.225.115  10.1.225.156     TCP        56 10000 → 2630 [ACK] Seq=6
 14675 397.553537172  10.1.225.156     140.127.225.115  TCP        62 2631 → 10000 [ACK] Seq=4

▸ Frame 11640: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 140.127.225.115, Dst: 10.1.225.156
▸ Transmission Control Protocol, Src Port: 10000, Dst Port: 2263, Seq: 1922, Ack: 2023, Len: 49
▸ Data (49 bytes)

0000  00 04 00 01 00 06 94 c6  91 20 04 3b 08 00 08 00   ········ ·;····
0010  45 00 00 59 ad 95 40 00  40 06 33 79 8c 7f e1 73   E··Y··@· @·3y···s
0020  0a 01 e1 9c 27 10 08 d7  4a b3 39 7b 8f 27 93 ac   ····'··· J·9{·'··
0030  50 18 01 f6 59 dc 00 00  30 2f 00 06 6d 65 74 65   P···Y··· 0/··mete
0040  72 41 7b 32 30 32 31 5f  31 30 5f 31 33 5f 32 30   rA{2021_ 10_13_20
0050  5f 34 34 5f 31 31 2c 20  6d 65 74 65 72 41 2c 20   _44_11,  meterA,
0060  33 30 2c 20 34 2e 32 33  7d                        30, 4.23 }
```

Fig. 8. (Color online) Unencrypted message intercepted by Wireshark.

the MQTT server within 1 s. The transmission speed is plotted against the amount of information in Fig. 9. One can observe that the greater the message content, the lower the transmission speed. Also, the transmission speed was higher for the unencrypted process than for the encrypted one. This may have resulted from the operation of the Galois/Counter Mode (GCM) block encryption.(17–19) In the GCM block encryption, the message authentication code (MAC) value

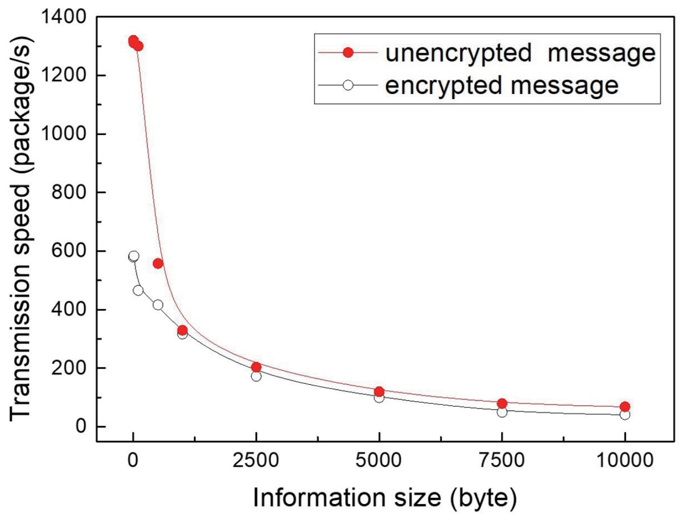


Fig. 9. (Color online) Transmission speeds of unencrypted (red dots) and encrypted (open black circles) messages for different amounts of information.

should be added to the message, which requires 16 bytes to verify the identity.(20) When the message content is small, adding the MAC value is more likely to affect the transmission speed, resulting in a long time for the encryption process. However, when the message content exceeds 1000 bytes, one can ignore this time delay.

Unfortunately, the ESP32 board cannot be equipped with anti-virus software owing to its architecture. If an attacker directly tampers with the internal code in the chip, it may be possible to steal most of the program or data. We recommend that the source code of the network devices be protected to prevent a third person from reading or modifying it.

4. Conclusion

In this study, we showed that a Hall sensor is suitable for the detection of water flow rate and leakage. A wireless flow meter based on the ESP32 board, combined with a Hall detector, can instantly receive and process the sensor data and then issue a pipeline leakage alarm, demonstrating the effectiveness of the flow meter for pipeline leakage detection. It is expected that this method can effectively reduce the leakage of water pipelines. We successfully demonstrated that IoT messages can be encrypted in MQTT transmission in the IoT field of information security. A safe and confidential transmission environment prevented third parties from intercepting or tampering with IoT devices during information transmission.

This study is in line with the United Nations Sustainable Development Goals and Taiwan's sustainable development targets and corresponding indicators for ensuring environmental quality and the sustainable management of ecological resources.

Acknowledgments

We are grateful for financial support from ROC MOST under Grant No. 110-2511-H-390-001.

References

1 CTCI Foundation: https://www.ctci.org.tw/ (accessed October 2021) (in Chinese).
2 National Development Council: https://www.ndc.gov.tw/ (accessed October 2021) (in Chinese).
3 D. C. Esty, L. Marc, S. Tanja, and S. Alexander de: Environmental Sustainability Index: Benchmarking National Environmental Stewardship (Yale Center for Environmental Law and Policy, New Haven 2005). https://doi.org/10.7927/H40V89R6
4 Taiwan Water Corporation: https://www.water.gov.tw/ch/Contents?nodeId=718 (accessed October 2021).
5 Taiwan Water Corporation: https://www.water.gov.tw/ch/Subject/Detail/1288?nodeId=813 (accessed October 2021).
6 Juniper Research: Number of IoT Devices to Triple by 2021: https://internetofbusiness.com/juniper-research-triple-iot-devices-2021/ (accessed October 2021).
7 H. Prihtiadi, A. Azwar, and M. Djamal: AIP Conf. Proc. **1719** (2016) 030045. https://doi.org/10.1063/1.4943740
8 R. F. Rahmat, I. S. Satria, B. Siregar, and R. Budiarto: IOP Conf. Ser.: Mater. Sci. Eng. **190** (2017) 012036. http://doi.org/10.1088/1757-899X/190/1/012036
9 J.-R. Lee and H. Tsuda: Opt. Lett. **30** (2005) 3293. https://doi.org/10.1364/OL.30.003293
10 J. P. Shri Tharanyaa, A. Satheesan, R. Saravanakumar, and D. Sharmila: IOP Conf. Ser.: Mater. Sci. Eng. **1084** (2021) 012122. http://doi.org/10.1088/1757-899X/1084/1/012122
11 J. Gautam, A. Chakrabarti, S. Agarwal, A. Singh, S. Gupta, and J. Singh: Water Supply **20** (2020) 1103. https://doi.org/10.2166/ws.2020.035
12 W. Stallings: Cryptography and Network Security Principle and Practice (Pearson, Upper Saddle River, NY, 2011) 5th ed., Chap. 16.
13 F. Aliyu, T. Sheltami, and E. M. Shakshuki: Procedia Comput. Sci. **141** (2018) 24. https://doi.org/10.1016/j.procs.2018.10.125
14 P. M. Desai, N. K. Zambare, K. R. Pawar, and Y. S. Patel: Int. J. Res. Electron. Comput. Eng. **6** (2018) 175.
15 Y. Sasaki and T. Yokotani: Technol. Innov. **4** (2019) 21.
16 National Security Agency: https://www.defense.gov/ (accessed October 2021). https://media.defense.gov/2021/Jan/05/2002560140/-1/-1/0/ELIMINATING_OBSOLETE_TLS_UOO197443-20.PDF
17 M. Pitchaiah, P. Daniel, and Praveen, Int. J. Sci. Eng. Res. **3** (2012) 1048. https://www.ijser.org/onlineResearchPaperViewer.aspx?Implementation-of-Advanced-Encryption-Standard-Algorithm.pdf
18 N. Ahmad, L. M. Wei, and M. H. Jabbar: J. Phys.: Conf. Series **1019** (2018) 012008. https://doi.org/10.1088/1742-6596/1019/1/012008
19 C. Paar and J. Pelzl: Understanding Cryptography (Springer, Heidelberg, 2009) Chap. 5. https://doi.org/10.1007/978-3-642-04101-3
20 C. Paar and J. Pelzl: Understanding Cryptography (Springer, Heidelberg, 2009) Chap. 12. https://doi.org/10.1007/978-3-642-04101-3

About the Authors

Chih-Chun Chang is a senior student currently studying in the Department of Applied Physics at the National University of Kaohsiung. His research focuses on physics education, Arduino electronics, and especially IoT applications. This publication is the first SCI journal paper he submitted.

Yao-Yu Lee is a senior student in the Department of Applied Physics at the National University of Kaohsiung. He's interested in using Arduino-related electronics to develop an Integrated microsensor.

Ting-Yu Hou is a junior student currently studying in the Department of Applied Physics at the National University of Kaohsiung. His research interests are physics, teaching aids made by Arduino electronics, and IoT technology.

Chin-Chung Yu received his Ph.D. degree in Physics from National Cheng Kung University, Taiwan, R.O.C., in 2000. Currently, he is an associate professor of the Department of Applied Physics, National University of Kaohsiung, and the President of the Physics Education Society of Taiwan, R.O.C. His current research interests are IoT applications, Physics education, crystalline films growth, and magnetic materials.