Sensors and Materials, Vol. 38, No. 11 (2026) 6557–6580
MYU Tokyo

S & M 4695

Design of a Chaos-based Cryptosystem for the Security of Sensed Data Transmission with Message Queuing Telemetry Transport-Based Industrial Internet of Things

Guo-Hsin Hu,[1] Wei-Zhe Xi,[2] and Jun-Juh Yan[2*]

[1]Department of Industrial Upgrading Service, Metal Industries Research & Development Centre, No. 1001, Kaonan Highway, Nanzih District, Kaohsiung 81160, Taiwan
[2]Department of Electronic Engineering, National Chin-Yi University of Technology, No. 57, Section 2, Zhongshan Road, Taiping District, Taichung 411030, Taiwan

(Received May 16, 2026; accepted July 23, 2026)

Keywords: chaos, industrial IoT, MQTT protocol, dynamic message topics

To address the sensed data transmission security in the Industrial IoT (IIoT), we propose in this paper a novel design of synchronized chaos-based dynamic keys and topics for developing a secure data transmission channel in Message Queuing Telemetry Transport (MQTT)-based IIoT. First, a real-time synchronization using multiple control inputs is newly proposed for master–slave discrete chaotic systems. This design, under the MQTT transmission protocol, does not increase the complexity of the synchronization controller or transmission time. Instead, it simplifies the design of synchronization controllers and significantly reduces the transient time required for synchronization, thereby enhancing the efficiency of secure data communication systems. The technological contributions of this paper include the design of real-time synchronization control, dynamic keys for cryptosystems, and synchronized dynamic message topics for MQTT-based IIoT. Finally, the developed technologies are integrated to construct an IIoT communication security system, which includes multiple robotic arms, demonstrating the security and feasibility of the designed system.

1. Introduction

In recent years, improvements in information and communication technology have popularized the applications of IoT, especially in Industry 4.0 automation. Through the convenient communication protocols of IoT, the Industrial IoT (IIoT) integrates the operational information of machines, enabling the collection of vast amounts of important sensed data. Coupled with artificial intelligence, this integration facilitates system modeling, analysis, and application, effectively reducing labor costs and significantly enhancing factory productivity.[1,2] Compared with wired network transmissions, wireless networks for data sharing in IoT architectures pose greater vulnerability to security attacks, challenging the security of data

*Corresponding author: e-mail: jjyan@ncut.edu.tw
https://doi.org/10.18494/SAM6442

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

transmission. Therefore, data protection becomes a crucial challenge when exchanging data in IoT using public wireless networks.[3] Message Queuing Telemetry Transport (MQTT) is a lightweight communication protocol commonly used in IoT, characterized by low network bandwidth and hardware resource requirements, making it well-suited for low-power and bandwidth-limited IoT environments.[4] In this study, the considered application scenario is explicitly restricted to a single-master, time-triggered polling-based industrial control architecture, in which a single master node periodically initiates communication cycles and multiple slave nodes respond accordingly. All theoretical derivations, synchronization mechanisms, security analyses, and experimental validations are conducted under this deterministic master–slave polling framework. Accordingly, the proposed framework should be interpreted as a secure communication solution for deterministic industrial automation systems operating under a master–slave polling mechanism, rather than as a general-purpose security architecture for arbitrary MQTT-based IIoT deployments. Within this framework, MQTT is employed solely to provide reliable message delivery, low-overhead communication for embedded devices, and simplified system deployment and maintenance. The publish–subscribe mechanism is used only to support the master-controlled communication flow. Nevertheless, even under this industrial architecture, field devices often possess limited computational resources and lightweight security mechanisms, making the confidentiality and integrity of transmitted sensing data susceptible to cyberattacks. Consequently, although MQTT-based IIoT significantly improves the efficiency and flexibility of industrial data collection, it also introduces increasing cybersecurity risks that must be carefully addressed.[5]

Conventional symmetric or asymmetric cryptosystems are commonly employed to ensure secure information transmission by utilizing static keys for encryption and decryption.[6] However, the emergence of quantum computing will pose a significant threat to traditional encryption techniques, and algorithms such as Shor and Grover potentially destroy the security.[7,8] In the near future, quantum computers with sufficient qubits can surpass current computational capabilities and endanger existing cryptosystems.[9,10] Additionally, the security of transmitting and storing static keys in symmetric encryption is also a critical problem in IIoT devices. Intercepted static keys during transmission can lead to the complete decryption of encrypted data. To solve this threat, we propose in this study a chaos-based synchronized dynamic key design to not only enable automatic key updates and synchronization between transmitter and receiver, but also resolve the distribution and transmission problem inherent in traditional encryption algorithms.

On the other hand, chaotic systems represent highly complex nonlinear systems that are characterized by dynamic behaviors that exhibit pseudo-randomness and aperiodicity. Moreover, the random-like state responses of chaotic systems differ from those of typical linear or nonlinear systems. Chaotic systems also feature specific strange attractors, within which states dynamically and randomly move without convergence or divergence. Furthermore, the state response of chaotic systems is considerably affected by initial values, which is referred to as the butterfly effect.[11] Utilizing these characteristics, chaotic systems have been extensively applied in the design of random number generators and data encryption required for communication security. In accordance with Refs. 12 to 20, we firstly construct a new 2D discrete hyper-chaotic

map with a linear cross-coupled topological structure combined with the Tent and Logistic map. The proposed map with the aforementioned structure enables it to outperform other enhanced chaotic maps developed recently. Secondly, an efficient PRNG based on the proposed one is implemented on the field-programmable gate array (FPGA explore chaos theory for generating high-quality random numbers and ensuring randomness using tests such as NIST and ENT). Prior works[15,16] utilize these numbers for generating dynamic chaotic keys but overlook chaos synchronization controls, leading to key discrepancies owing to the butterfly effect. To address this, researchers in chaotic cryptography[17–19] propose synchronization controllers to suppress the butterfly effect and ensure identical keys between transmitter and receiver. However, these methods overlook the transient time before synchronization, which significantly impacts encryption/decryption efficiency, warranting further discussion on effective control strategies to enhance performance. In this study, to effectively enhance the quality of dynamic keys and deduce the transient time of synchronized master–slave chaotic systems, a real-time synchronization design method for chaotic systems is proposed. This method combines hash functions (SHA3-256) and synchronizes chaotic random numbers to synchronously generate dynamically updated random keys at the subscriber and publisher of MQTT in IoT. This design, leveraging the characteristic of dynamic key updates, mitigates the possibility of brute-force attacks. In machine-to-machine networking, MQTT-based IIoT has gained traction for monitoring critical environmental data. However, the openness of the MQTT publish-subscribe model and the limitations of built-in authentication make it susceptible to attacks by intruders.[21,22] To enhance IIoT's scalability and security, we propose in this study a new design of MQTT dynamic message topics. By employing synchronized random number generators to create dynamic message topics, unauthorized access is thwarted to ensure data security. Integrating dynamic keys and topics into MQTT-based IIoT systems can guarantee data transmission security.

The subsequent sections of this study are structured as follows. In Sect. 2, we describe the main structure of MQTT-based IIoT systems and the problem formulation, including the discretization of continuous chaotic systems and the design of synchronization controllers. In Sect. 3, we present the design of random number generators by integrating synchronized chaotic states with SHA3-256. Additionally, statistical tests are employed to verify the randomness quality of the designed random sequences. In Sect. 4, we propose the encryption design of information in MQTT-based IIoT, including the design and implementation of dynamic message topics and synchronization controllers within the MQTT architecture, as well as the realization of demonstration systems. Finally, we conclude the study in Sect. 5.

2. Problem Formulation and System Architectures

As stated in the above research motivations, this study combines synchronized chaos-based dynamic keys with symmetric AES encryption algorithms and proposed MQTT dynamic topic design to establish the MQTT-based IIoT communication security system illustrated in Fig. 1. In Fig. 1, the types of machine can vary, including robotic arms or other machinery, and these machines may be distributed across different locations. Communication interfaces with the

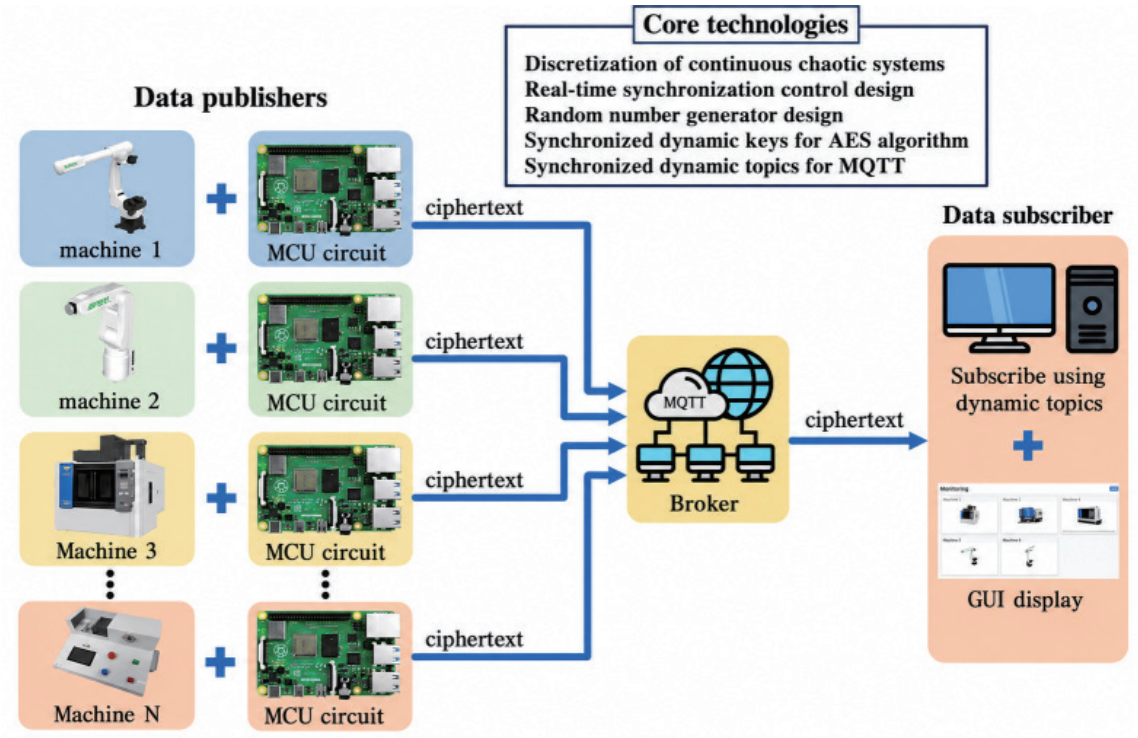


Fig. 1. (Color online) MQTT-based IIoT communication security system.

machines are established through microcontrollers to enable data reception. Additionally, a microcontroller also functions as the data encryption and publishing/subscribing system. After encrypting the data, it is securely transmitted via the MQTT v3.1.1 protocol to the monitoring endpoint. The secure transmission channel is realized through the real-time synchronization technology introduced in this paper. This control technology ensures synchronization between the master–slave chaotic systems embedded at the subscriber and publisher to facilitate the generation of synchronized dynamic keys and topics for encryption/decryption and publishing/subscribing. This IIoT communication security system encompasses various core technologies including the discretization of continuous-time chaotic systems, the design of real-time synchronization controllers, the implementation and performance analysis of MQTT-based IIoT chaotic dynamic key and topic realization, as well as the verification and implementation of security measures in the machine-to-machine communication system. The detailed descriptions of these technical aspects are elaborated below.

2.1. Discretization of continuous chaotic systems

Compared with digital circuits, analog circuits exhibit less stability in system parameters, rendering them susceptible to environmental factors and component aging. Furthermore, the chaotic behavior of continuous systems is significantly affected by system parameters, making them less suitable for encryption applications. Also, in remote wireless IIoT architectures, the

implementation of continuous chaotic systems using a digital microcontroller may encounter challenges in synchronization control between master and slave chaotic systems embedded at transmitter and receiver ends. The unpredictable transmission delays in wireless networks can disrupt synchronization control. To ensure synchronization, adopting discretized chaotic systems and designing synchronization controllers for discrete systems are necessary. To achieve this goal, the process of discretization for continuous chaotic systems is explained to preserve the chaotic behavior of the original continuous system. Generally, the dynamic equation for a continuous chaotic system is described as

$$\dot{x}(t) = Ax(t) + Hg(x(t)), \tag{1}$$

where $x(t) \in R^n$ and $g(x(t)) \in R^m$ are the state and nonlinear vectors, respectively. $A \in R^{n\times n}$ and $H \in R^{n\times m}$ are system matrices, and the matrix A is invertible. The discretized chaotic system corresponding to the continuous chaotic system (1) can be represented as[23]

$$x_d((k+1)T) = Gx_d(kT) + Wg(x_d(kT)), \tag{2}$$

where $G = e^{AT} \in R^{n\times n}$, $W = [G - I_n]A^{-1}H \in R^{n\times m}$, T represents the sampling time, and I_n is the identity matrix. The discretized chaotic system represented by Eq. (2) is obtained using an approximation method. Therefore, it still needs to check through a strange attractor, Lyapunov exponents, and the butterfly effect to confirm the existence of chaotic behavior. Using the Liu three-dimensional chaotic system as an example, we explain the process of discretization. Its dynamic equations are represented as

$$\begin{aligned} \dot{x}_1(t) &= a(x_3(t)) - x_2(t)x_3(t), \\ \dot{x}_2(t) &= bx_2(t) + x_1(t)x_3(t), \\ \dot{x}_3(t) &= -cx_3(t) - 4x_1(t)x_2(t), \end{aligned} \tag{3}$$

where a = 3.8, b = 2.5, and c = 7. Rearranging Eq. (3) into the form of Eq. (1), we obtain

$$\dot{x}(t) = \begin{bmatrix} -3.8 & 0 & 3.8 \\ 0 & 2.5 & 0 \\ 1 & 0 & -7 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -4 \end{bmatrix} \begin{bmatrix} x_2(t)x_3(t) \\ x_1(t)x_3(t) \\ x_1(t)x_2(t) \end{bmatrix}. \tag{4}$$

Utilizing $G = e^{AT}$ with a sampling time T = 0.001, W = [G − I_n]$A^{-1}H$, leads to the discrete Liu three-dimensional chaotic system, as shown in Eq. (5).

$$x_d((k+1)T) = Gx_d(kT) + Wg(x_d(kT)) \tag{5}$$

Here,

$$G=\begin{bmatrix} 0.9962 & 0 & 0.0038 \\ 0 & 1.0025 & 0 \\ 0 & 0 & 0.993 \end{bmatrix}, W=\begin{bmatrix} -0.001 & 0 & 0 \\ 0 & 0.001 & 0 \\ 1 & 0 & -0.004 \end{bmatrix}. \tag{6}$$

Expanding the derived discrete system (5), we obtain the following equations:

$$\begin{aligned} x_1\left((k+1)T\right) &= 0.9962x_1\left(kT\right)+0.0038x_3\left(kT\right)-0.001x_2\left(kT\right)x_3\left(kT\right), \\ x_2\left((k+1)T\right) &= 1.0025x_2\left(kT\right)+0.001x_1\left(kT\right)x_3\left(kT\right), \\ x_3\left((k+1)T\right) &= 0.993x_3\left(kT\right)-0.004x_1\left(kT\right)x_2\left(kT\right). \end{aligned} \tag{7}$$

To analyze and compare the discretized system (7) with the dynamics of the continuous system (4), we set the same initial values of systems by $\left(x_1(0),x_2(0),x_3(0)\right)$ = (12.2655, 10.3521, 12.1454). By MATLAB simulations, we obtain the chaotic strange attractors of the two systems, as shown in Figs. 2 and 3. In these figures, it can be observed that both the discretized

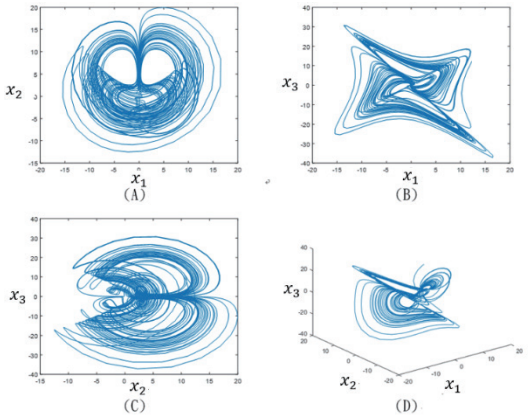


Fig. 2. (Color online) Strange attractor of the continuous chaotic system: (A) x_1,x_2, (B) x_1,x_3, (C) x_2,x_3, and (D) x_1,x_2,x_3.

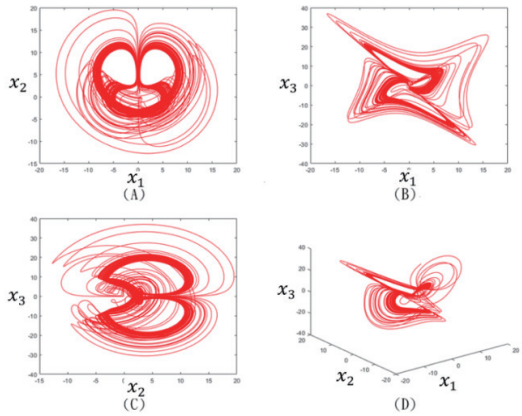


Fig. 3. (Color online) Strange attractor of the discrete chaotic system: (A) x_1,x_2, (B) x_1,x_3, (C) x_2,x_3, and (D) x_1,x_2,x_3.

and continuous chaotic systems exhibit chaotic strange attractors. Next, we calculate Lyapunov exponents[24] to assess the chaotic behavior of the discretized chaotic system (7). After calculation, the Lyapunov exponents of the system are obtained and shown in Table 1. It can be observed that all Lyapunov exponents are greater than 0, confirming that the discretized Liu chaotic system retains chaotic behavior. Hence, it can be utilized as a discretized chaotic system in subsequent designs as needed.

2.2. Design of real-time synchronization controller and analysis of error dynamics

In the traditional controller design, the design of multi-input control is less commonly used. This is because in the traditional analog circuit, multi-dimensional control inputs will increase circuit complexity and costs. However, in digital systems, controllers are implemented through program coding. Moreover, in the MQTT transmission protocol, the MQTT packet format has a message payload length of 2^{16} bytes. Such payload length allows for the transmission of multi-dimensional control signals and machine data in one go. Therefore, the design of multi-input control does not increase the circuit complexity or the transmission time in the MQTT-IIoT architecture. On the contrary, it simplifies the design of real-time synchronization and makes it feasible, which considerably reduces the transient time required for synchronization. This can effectively enhance the performance of communication security systems. Next, using the Liu discrete chaotic system as an example, we give the dynamic equations of the master and slave Liu discrete chaotic systems, as shown in Eqs. (8) and (9), respectively.

Master system:

$$\begin{aligned} x_1\left((k+1)T\right) &= 0.9962x_1\left(kT\right) + 0.0038x_3\left(kT\right) - 0.001x_2\left(kT\right)x_3\left(kT\right) \\ x_2\left((k+1)T\right) &= 1.0025x_2\left(kT\right) + 0.001x_1\left(kT\right)x_3\left(kT\right) \\ x_3\left((k+1)T\right) &= 0.993x_3\left(kT\right) - 0.004x_1\left(kT\right)x_2\left(kT\right) \end{aligned} \tag{8}$$

Slave system:

$$\begin{aligned} y_1\left((k+1)T\right) &= 0.9962y_1\left(kT\right) + 0.0038y_3\left(kT\right) - 0.001y_2\left(kT\right)y_3\left(kT\right) + u_1\left(kT\right) \\ y_2\left((k+1)T\right) &= 1.0025y_2\left(kT\right) + 0.001y_1\left(kT\right)y_3\left(kT\right) + u_2\left(kT\right) \\ y_3\left((k+1)T\right) &= 0.993y_3\left(kT\right) - 0.004y_1\left(kT\right)y_2\left(kT\right) + u_3\left(kT\right) \end{aligned} \tag{9}$$

Table 1
Lyapunov exponents of the discrete Liu chaotic system.

x_1	x_2	x_3
$\lambda = 0.2693$	$\lambda = 0.2065$	$\lambda = 0.2712$

In Eq. (9), $u_i(kT)$, $i = 1, 2, 3$ represents the control inputs to ensure that the state response of the slave system can synchronize with that of the master system, i.e., $y_i(kT) = x_i(kT)$, $i = 1, 2, 3$. The error vector between the master and slave systems is defined as $e_i(kT) = y_i(kT) - x_i(kT)$. Then, the error dynamic equations are

$$e_i\left((k+1)T\right) = f_i\left(kT\right) + u_i\left(kT\right), i = 1,2,3, \tag{10}$$

where

$$\begin{aligned} f_1\left(kT\right) &= 0.9962e_1\left(kT\right) + 0.0038e_3\left(kT\right) - 0.001\left(y_2\left(kT\right)y_3\left(kT\right) - x_2\left(kT\right)x_3\left(kT\right)\right), \\ f_2\left(kT\right) &= 1.0025e_2\left(kT\right) + 0.001\left(y_1\left(kT\right)y_3\left(kT\right) - x_1\left(kT\right)x_3\left(kT\right)\right), \\ f_3\left(kT\right) &= 0.993e_3\left(kT\right) - 0.004(y_1\left(kT\right)y_2\left(kT\right) - x_1\left(kT\right)x_2\left(kT\right)). \end{aligned} \tag{11}$$

Now, let the control input $u_i(kT)$ be defined as

$$u_i\left(kT\right) = -f_i\left(kT\right) + \alpha e_i\left(kT\right), \ \ |\alpha| < 1. \tag{12}$$

By substituting $u_i(kT)$ into Eq. (9), one has

$$e_i\left((k+1)T\right) = \alpha e_i\left(kT\right), \quad i = 1,2,3. \tag{13}$$

Since $|\alpha| < 1$ is selected in Eq. (12), $e_i(kT)$ is expected to converge to zero, which indicates that synchronization can be achieved. From Eq. (13), it is clear that the convergence speed of the synchronization error is dependent on the parameter α. When $|\alpha|$ approaches 0, the convergence speed increases. Specifically, when $\alpha = 0$, synchronization can be achieved in just one iteration period. An iteration is defined as a complete synchronization cycle consisting of transmission of control information, successful reception at the slave node, state update computation, and preparation for the next synchronization cycle. Synchronization is therefore event-driven and not dependent on a fixed sampling period. Therefore, this iteration-based synchronization will be used in this paper to design real-time synchronization controllers, accelerating the synchronization speed between the master and slave systems, eliminating synchronization transient time, and enhancing system performance. In the following, we discuss the effect of the parameter α on the synchronization speed. In the simulation, the initial values of the master and slave systems are set as $(x_1(0), x_2(0), x_3(0)) = (12.2655, 10.3521, 12.1454)$, $(y_1(0), y_2(0), y_3(0)) = (1.3228, 5.4556, -8.2562)$. We analyze three different parameter settings, $\alpha = 0.8$, $\alpha = 0.5$, and $\alpha = 0$, to observe the relationship between α and the convergence of the error state. The results are shown in Fig. 4, where it can be observed that as $|\alpha|$ approaches zero, the synchronization speed increases.

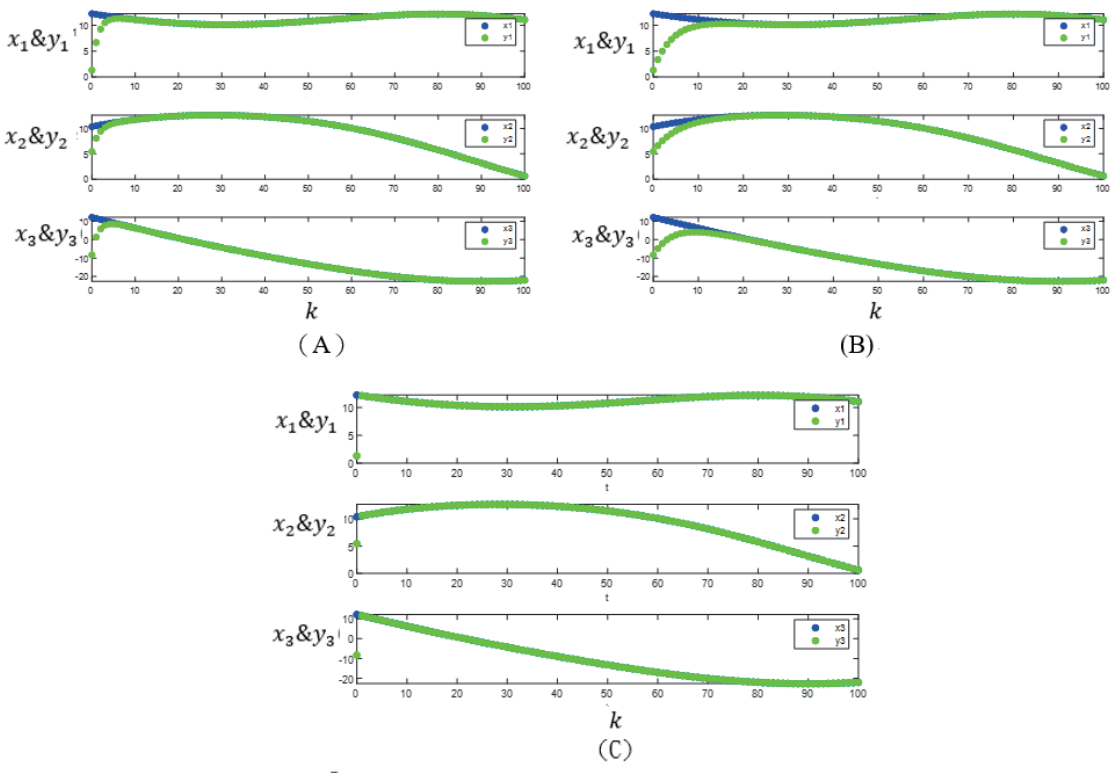


Fig. 4. (Color online) Convergence speed of error state: (A) $\alpha = 0.8$, (B) $\alpha = 0.5$, and (C) $\alpha = 0$.

Further observation of the error dynamics (12) reveals that if we use $u_i(kT) = -f_i(kT) + \alpha e_i(kT)$, $i = 2,3$ with $\alpha = 0$, i.e., with dual control inputs, the error states $e_2(kT) = e_3(kT) = 0$, for $k1$, achieving real-time synchronization. Additionally, when $e_2(kT) = e_3(kT) = 0$ (i.e., $x_2(kT) = y_2(kT)$; $x_2(kT) = y_2(kT)$), it can be deduced that $e_1((k + 1)T) = 0.9962e_1(kT))$, indicating that $e_1(kT)$ will also converge to zero. Therefore, if only two synchronized chaotic states are required, dual control inputs can be employed. The response of the system with dual control inputs is depicted in Fig. 5.

3. Synchronization of MQTT-IIoT Chaotic Dynamic Keys

3.1. Implementation of synchronized MQTT-IIoT Chaotic Dynamic Keys

As mentioned above, after completing the real-time synchronization control, we propose the design of synchronized dynamic keys under the MQTT communication protocol, as shown in Fig. 6. Since only two synchronized chaotic states are needed to generate the dynamic keys and MQTT dynamic message topics, the synchronization controller in Fig. 6 can use the dual-input design proposed in the previous section. Considering the implementation of the system, it is evident that we need to divide the controller $u_i(kT)$ into control input components generated by

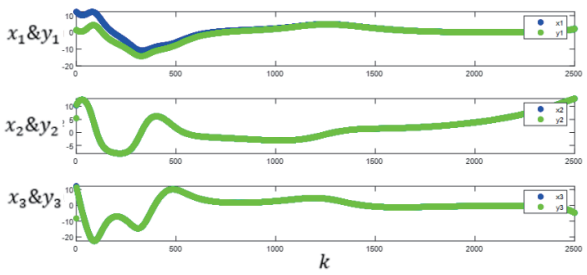


Fig. 5. (Color online) State response of chaotic system with dual-input signals.

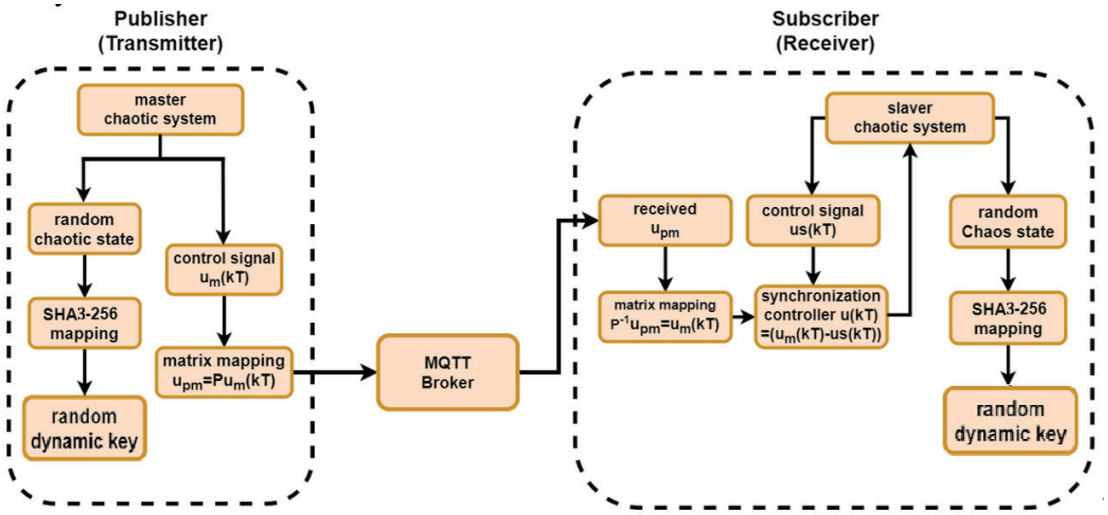


Fig. 6. (Color online) Diagram of the synchronized MQTT-IIoT chaotic dynamic keys.

the master and slave systems, denoted as $u_{mi}(kT)$ and $u_{si}(kT)$, respectively, where i = 2,3. The components $u_{mi}(kT)$ are generated and transmitted by the transmitter, while the receiver receives $u_{mi}(kT)$ through MQTT subscription. Then, by combining $u_{mi}(kT)$ with $u_{si}(kT)$ using the operation of $u_i(kT) = u_{mi}(kT) - u_{si}(kT)$, the complete control signal $u_i(kT)$ is formed to achieve synchronization at the subscriber. From Eqs. (10) to (12), we can derive

$$u_{m2}(kT) = 1.0025x_2(kT) + 0.001x_1(kT)x_3(kT) + \alpha x_2(kT), \tag{14}$$

$$u_{m3}(kT) = 0.993x_3(kT) - 0.004x_1(kT)x_2(kT) + \alpha x_3(kT), \tag{15}$$

$$u_{s2}(kT) = 1.0025y_2(kT) + 0.001y_1(kT)y_3(kT) + \alpha y_2(kT), \tag{16}$$

$$u_{s3}(kT) = 0.993y_3(kT) - 0.004y_1(kT)y_2(kT) + \alpha y_3(kT). \tag{17}$$

For $u_{mi}(kT)$, $i = 2,3$, additional protection can be achieved by further selecting an appropriate nonsingular matrix $P \in R^{2\times2}$. Before transmission, the matrix operation $u_{pm}(kT) = Pu_m(kT)$ can be performed. Upon receiving the data, the subscriber performs the operation $P^{-1}u_{pm}(kT)$ to obtain $u_{mi}(kT)$, $i = 2,3$. Consequently, the control input $u_i(kT) = u_{mi}(kT) - u_{si}(kT)$ can be completed at the subscriber. Therefore, the master–slave chaotic systems can achieve synchronization, and both the publisher and the subscriber can synchronously have identical chaotic random numbers. By SHA3-256 mapping, dynamic 256-bit random keys can be generated simultaneously, facilitating the integration with any symmetric encryption algorithm, such as AES, for data encryption applications. Subsequently, the Python program was designed with $\alpha = 0$ to implement the real-time synchronization using dual-input control. The results are presented in Fig. 7, illustrating the state of the master and slave chaotic systems and the dynamic keys over four sampling intervals. The left side in Fig. 7 represents the master system, whereas the right side represents the slave system. Figure 7 shows the chaotic states of the master and slave systems as well as the dynamic keys after SHA3-256. From the results, as expected, the chaotic states are $x_2(kT)$ and $y_2(kT)$; $x_3(kT)$ and $y_3(kT)$ of the master and slave systems synchronize completely after one sampling interval, and after SHA3-256, they can dynamically generate identical keys.

3.2. Randomness quality analysis of chaotic dynamic keys

The synchronized dynamic random numbers shown in Fig. 6 will be used for dynamic key encryption algorithms. Therefore, the randomness of these chaotic dynamic random numbers is crucial. Statistical analysis using NIST and ENT[25,26] will be conducted to verify their randomness quality and ensure the security of the system during application.

```
round 0
master starting state: [12.2655 10.3521 12.1454]
sha(x1): 52b9a562d8efa5a4b3da1e87ffecd3a6c2baa3c9c029664157493a9242275a19
sha(x2): b1dec052874a9f33e54b1c06559b8901e21be6f8f0efbeb043bf60ec2e4ca904
sha(x3): c32e914ed0d6e754d996cd82d63eb718d95a160eaa2f8b98ca68aad086fa3b86
round 1
master: [12.13939172 10.5271684  11.55455783]
sha(x1): 5f0dd29b8b4044bb3f9647af0f7e88236bd9f560bf9c25b0927c4d50ca6a2754
sha(x2): 1868ea98fef3226e7930af6ac6c9aef398bbecdc80629802c3207bba74cbe799
sha(x3): a2643693e5173d6f84dd28b1ce57cdb7a8a822bb8b755b270300eff39b661f95
round 2
master: [12.01561449 10.69396003 10.96456961]
sha(x1): 27a5e37712822bb92e704a83387d5f635fd4f908f9e63a089de75a7bf0ab5146
sha(x2): 1d39925fc7ca1a1013e69e9573806f0cea23b3e44c289b52f8aeba17763f775f
sha(x3): bcbea7d9442863c75de10b00a936f547dff00b198c6693509d8b02cdb5f32aea
round 3
master: [11.89445063 10.85263923 10.37590236]
sha(x1): cc2b002426d9ba4fb13f1a0d5cfdd4a803a4f4b190f84d48847e4da4c7db5b10
sha(x2): 399bafafa55dfa452b567ab37e81c3fc17b35f5a48f2ae634a82b3718e27cb9e
sha(x3): 285b36442220c0405424d71b313d500cecae0d0a6f55d2faa02e3c3b0aee8e11
round 4
master: [11.77616134 11.00337483  9.78898293]
sha(x1): 2377750b4079a8732890c0812ce501054c660f42f1b00b42b4ff14e613342b63
sha(x2): 95bbade96adc6f26520e5cb1b80ab75b40ad9ebd3241dc367552b3e81bc848b1
sha(x3): 84aed27fc2132e70d0a4bd11e59d4a83aeab7e58f920dcfe9a1b82beff3c58a3
```

```
round 0
slave starting state: [5.3591, -1.2369, 18.8424]
sha(y1): 253397492cd6c856b21c637156f71c28c655fedf9362f65808c5964afc3e80aa
sha(y2): 8bd934038a4958fb9254bf77d333341d132a15461bf6129defbfd376ee74fd8d
sha(y3): fb4ece9393fd7a5b782354c0dfc83242823ab1276df888f223127a18d1e86049
round 1
slave: [ 5.43325155 10.5271684  11.55455783]
sha(y1): 9428f705c5b6094b75fe57b3bae0f2a2f3956919f0a10a6f249b523a54b2e5dd
sha(y2): 1868ea98fef3226e7930af6ac6c9aef398bbecdc80629802c3207bba74cbe799
sha(y3): a2643693e5173d6f84dd28b1ce57cdb7a8a822bb8b755b270300eff39b661f95
round 2
slave: [ 5.23694002 10.69396003 10.96456961]
sha(y1): f64438bde30e0d319ce5aa8ee87f122346228890ab19e79f92a0bd65f632498d
sha(y2): 1d39925fc7ca1a1013e69e9573806f0cea23b3e44c289b52f8aeba17763f775f
sha(y3): bcbea7d9442863c75de10b00a936f547dff00b198c6693509d8b02cdb5f32aea
round 3
slave: [ 5.0463945  10.85263923 10.37590236]
sha(y1): 84889088777675a1e423f07fd12e8f6b3f32a964f542d8c9378a07788685c197e
sha(y2): 399bafafa55dfa452b567ab37e81c3fc17b35f5a48f2ae634a82b3718e27cb9e
sha(y3): 285b36442220c0405424d71b313d500cecae0d0a6f55d2faa02e3c3b0aee8e11
round 4
slave: [ 4.86211254 11.00337483  9.78898293]
sha(y1): abcaf15e784bd9e91701c7b58670a90018f23f2e68e6e8fe246f16f2023134d6
sha(y2): 95bbade96adc6f26520e5cb1b80ab75b40ad9ebd3241dc367552b3e81bc848b1
sha(y3): 84aed27fc2132e70d0a4bd11e59d4a83aeab7e58f920dcfe9a1b82beff3c58a3
```

Fig. 7. (Color online) Chaotic dynamic key generator and synchronization.

3.2.1. NIST test

NIST provides the SP 800-22 standard for random number testing. This test consists of 15 different criteria, and the test results are indicated by p-values. A p-value greater than 0.001 indicates a qualified result. In this paper, the dynamic random number generator will undergo randomness testing using the NIST SP800-22 standard. A set of binary data consisting of 7360000 bytes generated by the dynamic random number generator with initial values ($x_1(0)$, $x_2(0)$, $x_3(0)$) = (12.2655, 10.3521, 12.1454) will be tested. The results are shown in Table 2. According to the results in Table 2, the obtained random numbers pass all 15 tests, with a total score of 7.207383816623124. Therefore, it can be confirmed that the random number generator designed exhibits excellent randomness quality.

3.2.2. ENT test

Next, the ENT testing standard is used to analyze the randomness quality of the random numbers. The ENT test consists of five test items, and the results are shown in Table 3. In the ENT test standard, entropy analysis evaluates randomness by calculating the entropy value of the random sequence. A higher entropy value closer to 8 indicates higher randomness. The chi-squared test (x^2 distribution) evaluates whether the distribution of values is uniform, with the ideal result being between 10 and 90%. The arithmetic mean value test checks if the mean value exceeds 127.5, indicating higher randomness. The Monte Carlo value for Pi evaluates the

Table 2
Results of NIST SP800-22 test.

Statistical test	P-value (N = 7360000 bytes)	
Initial value	(12.2655, 10.3521, 12.1454)	
Monobit	0.4519213243113937	Pass
Block frequency	0.8319020634402219	Pass
Runs test	0.48382683206390265	Pass
Longest-run-of-ones in a block	0.4579365859840238	Pass
Binary matrix rank	0.9367982423320209	Pass
Discrete Fourier spectral	0.28770670956397354	Pass
Non-overlapping template matching	1.2830330957459524	Pass
Overlapping template matching	0.8138594502449501	Pass
Maurer's universal statistical	0.4493895715071338	Pass
Linear complexity	0.21691807570066063	Pass
Serial	0.011615924372179419	Pass
Approximate entropy	0.025249058826220073	Pass
Cumulative sums	0.4963517703192688	Pass
Random excursions	0.40334853500233325	Pass
Random excursions variant	0.05752657720888836	Pass
Total score	7.207383816623124	

Table 3
Results of ENT test.

Statistical test	Value (7360000 bytes)
Initial value	(12.2655, 10.3521, 12.1454)
Entropy	7.999978 bits per byte
x^2 Distribution	Chi square distribution for 7360000 samples is 222.63 and randomly would exceed this value 92.91% of the time
Arithmetic mean	Arithmetic mean value of data bytes is 127.5475
Monte Carlo π estimation	Monte Carlo value for Pi is 3.140714750
Serial correlation	Serial correlation coefficient is 0.000064

randomness by comparing the calculated value to π. The serial correlation coefficients test assesses the correlation between random numbers, with a closer result to 0 indicating higher randomness. On the basis of Table 3 and the aforementioned ENT test standard, although the result of the chi-squared test (x^2 distribution) is 92.91%, observing the results of other tests still confirms that the random numbers designed in this study exhibit good randomness behavior.

3.3. Implementation of synchronized dynamic keys under the MQTT-IIoT

In the following, we conduct the time analysis of synchronized dynamic keys. The realized test system is illustrated in Fig. 8. We utilized discrete master–slave chaotic systems represented by Eqs. (8) and (9), along with a real-time synchronization controller as Eq. (12) by setting $\alpha = 0$. The broker used in this experiment was Synology DiskStation DS420+, equipped with an Intel Celeron J4025 processor (2-core 2.0 / 2.9 GHz) and 2 GB of DDR4 non-ECC RAM. Both the publisher and subscriber were deployed on Raspberry Pi 4 (Model B 4 GB of RAM, 1.5 GHz quad-core Arm® Cortex®-A72 CPU). In the test program, we design a synchronization status flag. When the master–slave chaotic systems synchronized, the synchronization status flag was set to Low; otherwise, it was set to High. This facilitated the observation of the synchronization time after each synchronization, with a 1×10^{-6} s delay inserted repeatedly after recalculating with different initial values. The test results are depicted in Fig. 9, which shows that the first



Fig. 8. (Color online) System for measuring synchronization time under MQTT architecture.

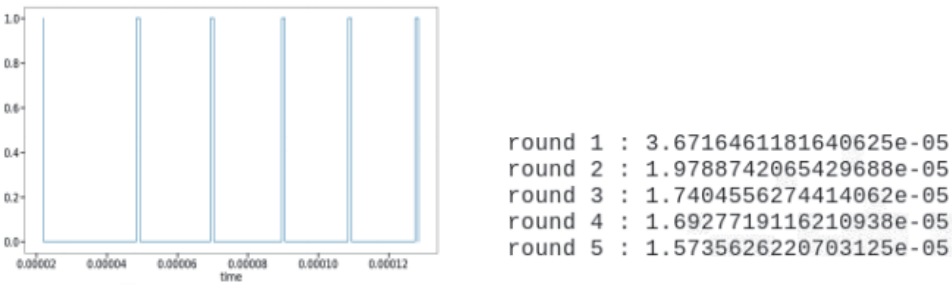


Fig. 9. (Color online) Synchronization time under MQTT-IIoT architecture.

synchronization time was relatively longer owing to the initial system boot settings. Additionally, slight differences in network transmission conditions may result in minor variations in synchronization time. From the results in Fig. 9, it is observed that, apart from the first synchronization, the synchronization time is approximately 1.9×10^{-5} s. This synchronization time specification is sufficiently fast for the application to collect and transmit machine data.

4. Technology Integration and Realization of MQTT-IIoT Secure Communication System

On the basis of the discussions in the previous sections, once the feasibility of chaos dynamic keys and the qualified synchronization performance under the MQTT-IIoT environment is confirmed, we can proceed with implementing the IIoT secure communication system, as depicted in Fig. 1. To realize this secure transmission system, in addition to the key generation and real-time synchronization discussed in the previous sections, some innovative designs are also proposed, including the innovative MQTT dynamic message topics, the data acquisition interface of the machines (publishers), and the GUI design for the subscribers. In the following, we provide the details of these designs.

4.1. Innovative design of MQTT dynamic message topics

MQTT operates on the basis of TCP/IP, but its transmission protocol differs from that of hypertext transfer protocol (HTTP). MQTT headers are encoded numerically with a total length of only two bytes, and following this header are the message topic and payload. The MQTT transmission protocol is based on the "publish/subscribe" mechanism. The MQTT message publisher acts as a data provider. The transmission is not directly sent to the subscriber but is coordinated through a MQTT broker. Each file published by a publisher has a unique static topic. The MQTT architecture offers advantages such as light weight and ease of use. However, without robust security mechanisms and knowing the static topic allow anyone to subscribe data from the MQTT broker and result in significant security risks. Currently, basic protections involve simple and fixed password encryption for MQTT brokers to prevent unauthorized access. However, if MQTT broker passwords are compromised and topics are stolen, the transmitted data can be easily accessed by hackers. To enhance the security of data transmission under the MQTT framework, we innovatively introduce dynamic message topics established using synchronized chaotic random numbers. Chaotic random numbers, being quasi-random signals, are unpredictable and can ensure that data remains undisclosed to unauthorized subscribers. The IIoT secure communication system designed is illustrated in Fig. 10. To implement the data security transmission architecture, the program flow is illustrated in Fig. 11. In Fig. 11, both the subscriber and the publisher initiate the program. The publisher selects the text file to be published and temporarily stores its content. Moreover, the publisher uses chaotic random numbers and expands them into two 256-bit random sequences using SHA3-256. These sequences are placed in the transmission packet as a dynamic message topic and AES keys. After encrypting the original file data, the encrypted file is transmitted to the MQTT broker.

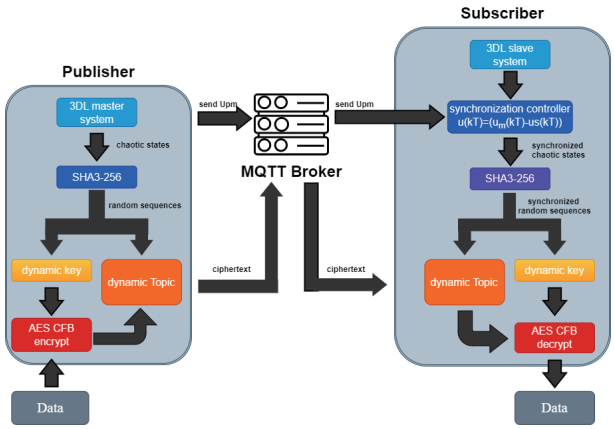


Fig. 10. (Color online) Data security transmission architecture.

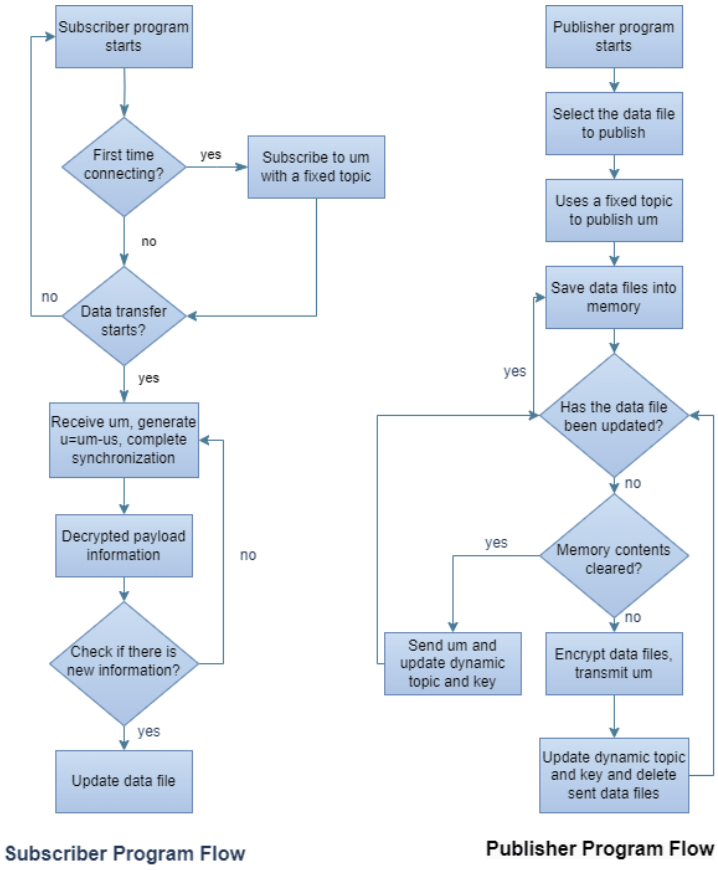


Fig. 11. (Color online) Program flowchart for data security system based on MQTT-IIoT.

Additionally, $u_{mi}(kT)$ are transmitted to the subscriber. This allows for the synthesis of $u_i(kT) = u_{mi}(kT) - u_{si}(kT)$. As discussed earlier, this ensures the synchronization. The dynamic message topic is then used to subscribe. When the system is first started, the publisher and subscriber perform a common fixed message topic publication and subscription to obtain the initial $u_{mi}(kT)$

and achieve the initial synchronization between the main and slave systems. After the synchronization, this fixed message topic is replaced by the dynamic message topics to enhance system security.

Remark 1: The system initialization stage is modeled as a transient synchronization phase, during which a temporary fixed-topic communication structure is used to establish initial coordination between the master and slave nodes. In this phase, the adversary is assumed to have only passive eavesdropping capability, without the ability to modify transmitted messages or interfere with internal system states. During this bootstrapping phase, only the master control signal $u_{mi}(kT)$ is transmitted over the communication channel, whereas the slave-side control input $u_{mi}(kT)$ is locally generated and remains undisclosed. Consequently, the attacker cannot reconstruct the complete control input $u_i(kT) = u_{mi}(kT) - u_{si}(kT)$, which prevents the inference of the internal synchronization dynamics. Although a fixed topic is temporarily used during initialization, the absence of full control information ensures that the adversary cannot obtain sufficient data to reconstruct subsequent chaotic states or derive the dynamically generated cryptographic keys. After synchronization is achieved, the system immediately transitions to a chaos-driven dynamic topic and dynamic key generation mechanism, thereby eliminating the persistent exposure risk associated with fixed communication structures. Therefore, the initialization phase does not leak sufficient information to compromise system security, since the transmitted signal $u_{mi}(kT)$ alone is insufficient to infer the complete control dynamics, and the locally generated slave control input further prevents the reconstruction of $u_i(kT)$ and the associated internal state evolution. As a result, the bootstrapping phase remains secure under a passive eavesdropping threat model.

Remark 2: To characterize robustness under network-induced delays, we first define an iteration-based synchronization framework. Accordingly, the synchronization process is event-driven rather than governed by a fixed sampling period. Under the proposed mechanism, synchronization is not dependent on a fixed sampling period or strict timing constraints. Instead, each update is triggered upon the successful completion of a communication event. As a result, network delays or jitters affect only the duration of an individual iteration, while the synchronization mechanism itself remains unchanged. Under bounded communication delays, the synchronization process proceeds through successive communication iterations, allowing synchronization errors to be corrected after each successful message exchange. Furthermore, the parameter α is introduced to characterize the synchronization speed. A smaller α value results in faster convergence. The practical implementation does not require synchronization to be completed within a fixed sampling period. Instead, $\alpha = 0$ is interpreted as an idealized design condition representing an instantaneous convergence tendency, whereas the actual system operates under an iteration-based execution model. Therefore, the overall stability and convergence of the proposed framework rely on the successful completion of successive communication iterations rather than on strict time-slot synchronization.

4.2. Implementation of MQTT-IIoT secure communication system

Integrating the developed technologies mentioned above, we now realize the data secure collection and remote monitoring of machinery during operation as shown in Fig. 12. In Fig. 12, the system utilizes Raspberry Pi 4B; as depicted in Fig. 13, the hardware specifications include 4 GB of RAM, utilizing Broadcom 2711 quad-core Cortex-A72 64-bit SoC with a single-core clock speed of 1.5 GHz. The Raspberry Pi serves as the publisher, which features a built-in master chaotic system responsible for data encryption and transmission to the broker. It can receive the machine information via an RS232 to USB serial interface. The machine employed is the HIWIN RA605-710-GC six-axis articulated robot arm, as shown in Fig. 13(b). It is connected

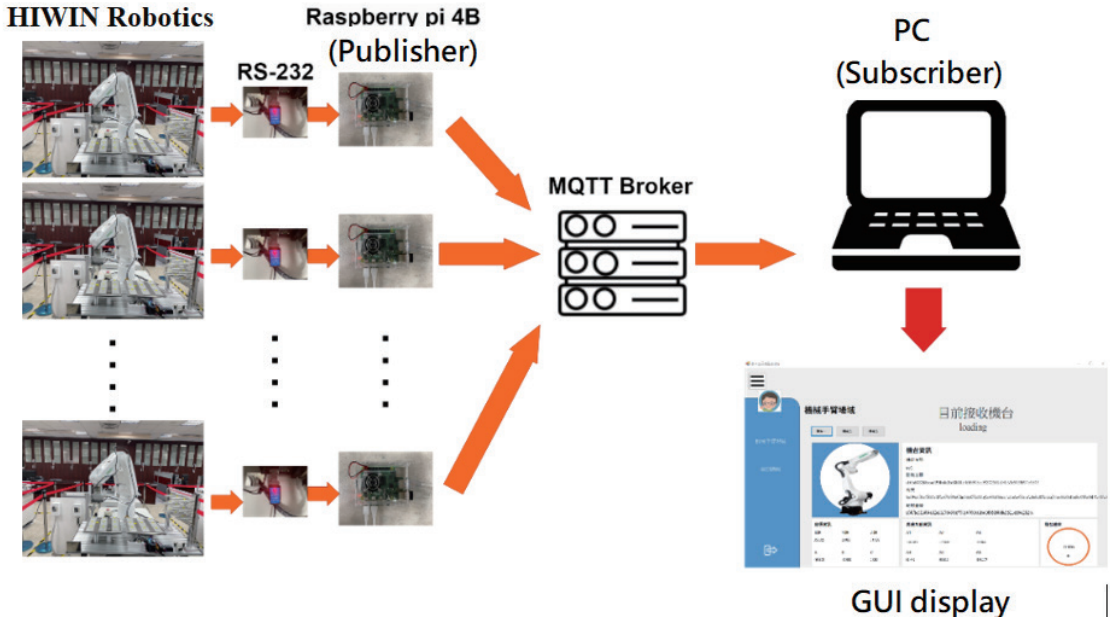
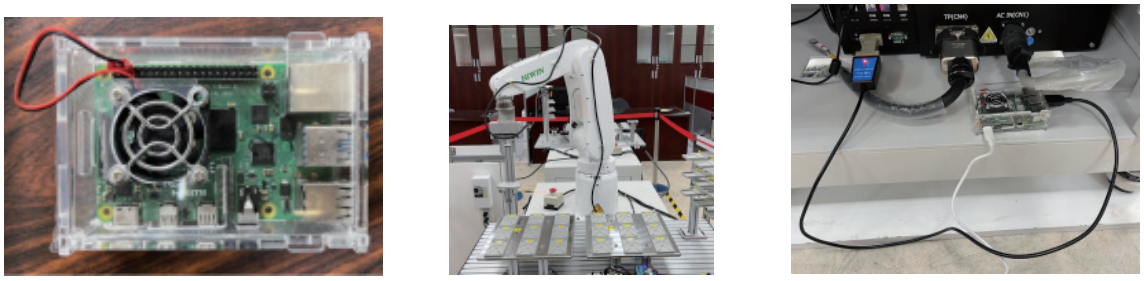


Fig. 12. (Color online) Realization of MQTT-IIoT secure communication system.



(a) (b) (c)

Fig. 13. (Color online) (a) Raspberry Pi 4B. (b) HIWIN RA605-710-GC robot arm. (c) RS232 to USB with Raspberry Pi 4B.

to Raspberry Pi 4B via RS232 to USB, as illustrated in Fig. 13(c), for acquiring the operation information. The central control computer of the robotic arm executes automation programs to control arm movements and send information such as gripper coordinates and motor angles to Raspberry Pi 4B. The obtained data is encrypted and transmitted to the broker using the MQTT v3.1.1 transmission protocol. This setup completes the architecture for machinery expansion, characterized by its small size, convenient installation, and flexibility for adjusting to industry demand.

In response to the increasing number of machines, it is essential to equip the system with the expanded mechanism for machines. Therefore, in this paper, we present the architectural diagram for the transmission of data among multiple machines, as depicted in Fig. 14.

In Fig. 14, data transmission is carried out through the MQTT protocol. When transmitting data among multiple machines, the computer (subscriber) first actively detects the names of Raspberry Pi 4B configurations associated with all machines through subscription. These names are then stored on the computer (subscriber) (as indicated by the blue arrows in Fig. 14). Subsequently, the computer adopts a polling method to publish data transmission requests on the basis of machine names. Raspberry Pi 4B confirms whether to transmit machine data to the computer by subscribing to the configured machine names (as indicated by the green arrows in Fig. 14). When data transmission is required, the MQTT secure transmission technology designed in this paper is used for the secure transmission of machine data. Consequently, the computer end can successfully receive data from the specified machine (as indicated by the red

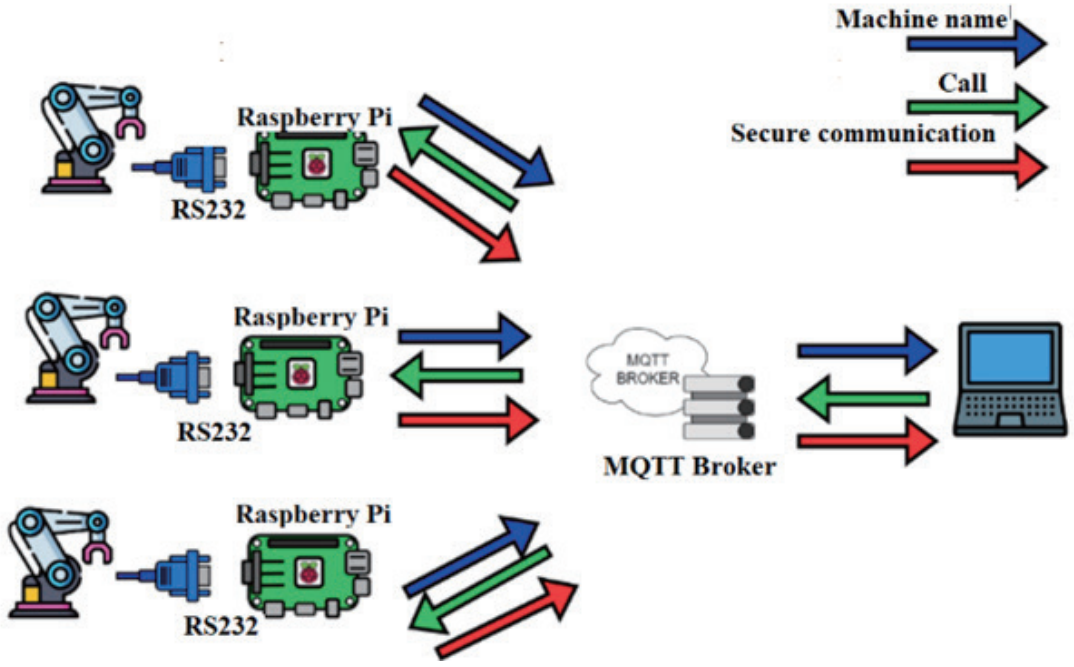


Fig. 14. (Color online) Diagram of multi-machine data transmission architecture.

arrows in Fig. 14). The operational workflow diagram of the computer in Fig. 14 is illustrated in Fig. 15. In Fig. 15, the computer adopts a multi-threaded design. One thread is primarily used to receive machine names and call on machines to send data. It continuously detects if new machines join and uses a time polling to determine which machine to call for data transmission. Another thread is responsible for switching between machines based on time. Machine time settings can be configured before the system is initiated. When the set time is reached, the time variable increases. Once the time variable exceeds the number of machine names, it resets to zero. A separate thread executes the previously designed secure transmission to retrieve, decrypt, and store data.

Figure 16 depicts the system flowchart of the Raspberry Pi 4B publisher established on each machine. At the system's onset, the machine's name needs to be inputted. Subsequently, the machine name is published, and the machine data is read. If no data is available, the reading process repeats. Once data is obtained, it is stored in a LIST sequence, which is then documented. Subsequently, the system subscribes to the topic of the machine name to confirm whether the computer terminal is calling the machine. If there is no call from the computer terminal, the process returns to publishing the machine name. However, when the computer terminal calls,

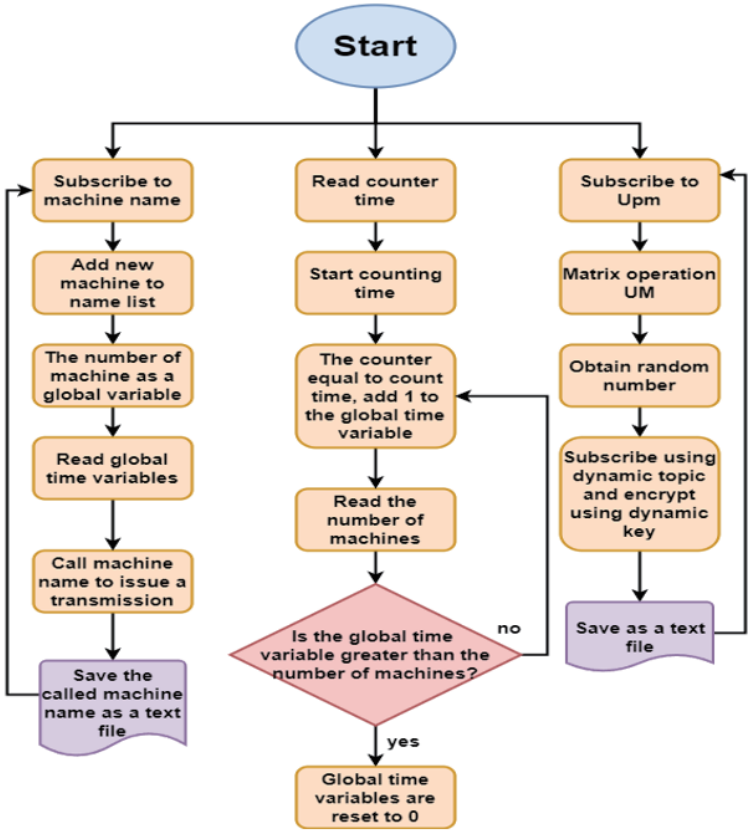


Fig. 15. (Color online) Operational workflow diagram of the computer.

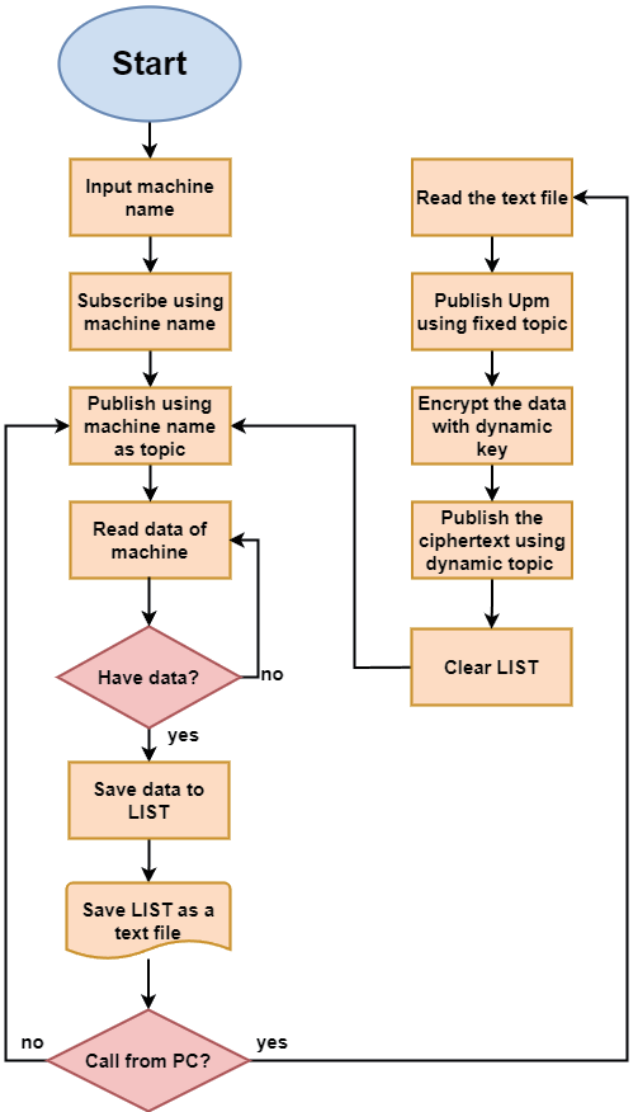


Fig. 16. (Color online) Raspberry Pi 4B (publisher) flow chart of data transmission.

the system reads the document from the LIST sequence, securely transmits it to the computer terminal via the MQTT transmission channel, clears the LIST sequence, and avoids redundant data transmission. This design allows for the transmission of machine data when a sufficient amount is collected, reducing the burden on network bandwidth, particularly in scenarios with numerous machines in the IIoT network. Additionally, it eliminates the need to modify program content when adding new machines; users can execute the program, only requiring different machine names.

GUI Monitoring Interface: At the computer terminal in Fig. 12, the GUI monitoring window is designed. The programming language used is C#. The designed application offers an operational interface, creating a comprehensive remote monitoring system applicable across

various domains and multiple machines. The GUI is illustrated in Fig. 17. In Fig. 17, the GUI includes machine switching buttons designed specifically for the three robotic arms used in this system. These buttons enable users to switch between machine windows freely to view the latest data information for each machine. Additionally, users can view the name of the machine currently being called by the computer (subscriber) and the data received. At the top of the GUI, database buttons are also provided. Clicking on these buttons opens the folder where the data is stored. Inside the folder, users can view data saved by date and machine name to facilitate users in utilizing the data effectively. Finally, the implementation of the MQTT-IIoT Secure Communication System is completed. For detailed system operation, please visit https://youtu.be/gGjx3peqiLE.

On the basis of the analysis presented in the previous sections and the results of the system implementation, the main contributions of this paper are summarized as follows:

Proposed a real-time synchronization design to enhance system efficiency:

In this study, we utilized a multi-input control approach to achieve real-time chaotic synchronization between master and slave systems. This significantly reduces the synchronization time required in chaotic systems, thereby improving the overall performance of the secure communication system and better meeting the practical demands of industrial IoT applications.

Integrated chaotic synchronization with an authentication mechanism to strengthen key management security:

By applying chaotic synchronization technology to the identity authentication process, the proposed method effectively addresses the challenges of key transmission and distribution

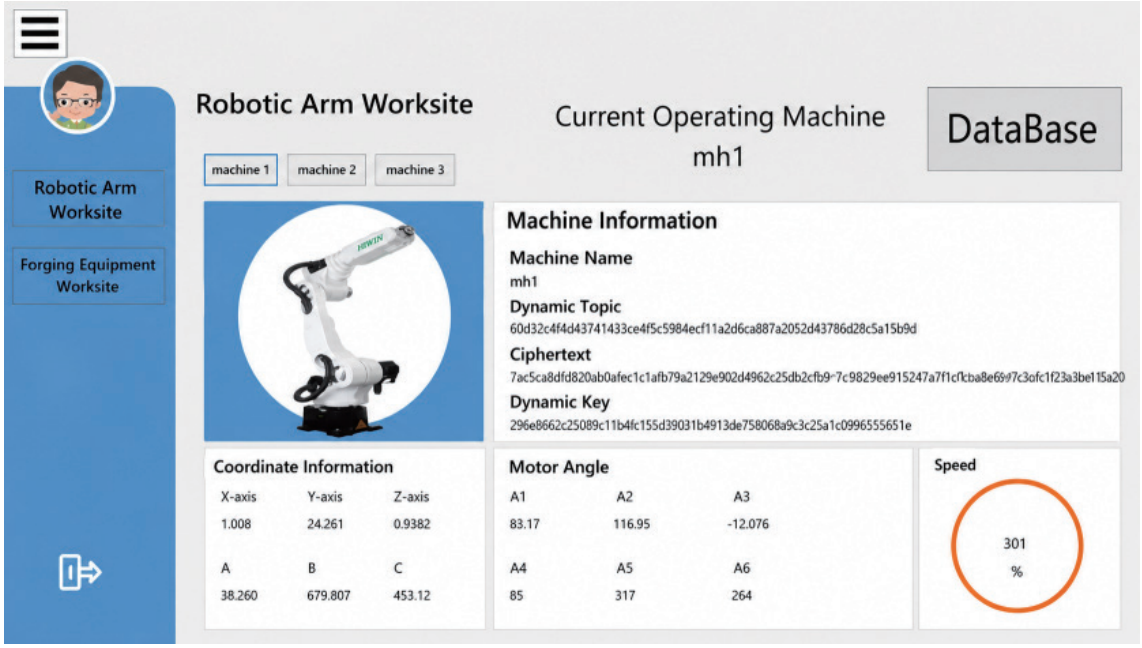


Fig. 17. (Color online) GUI monitoring window at the computer terminal.

found in traditional symmetric encryption. This enhances the reliability of mutual authentication and the confidentiality of data exchange.

Combined dynamically synchronized keys with AES encryption to improve data encryption strength:

By leveraging the high sensitivity and unpredictability of chaotic systems, the proposed method dynamically generates encryption keys that are integrated with the AES algorithm. Compared with traditional static keys, this dynamic key approach offers enhanced security and effectively mitigates the risk of brute-force attacks.

Proposed the dynamic topic design for MQTT to improve message transmission security:

A dynamic topic publishing mechanism based on random chaotic states is proposed, allowing only authorized devices that have completed synchronization verification to subscribe to data. This mechanism effectively prevents unauthorized entities from intercepting or accessing confidential information.

5. Conclusion

In this study, we integrated chaotic systems with real-time synchronization control to develop a chaos-based random number generator. The generated sequences were validated using standard statistical tests, including the NIST SP 800-22 and ENT test suites, to ensure their randomness quality. Furthermore, the proposed dynamic random number generator was incorporated into a symmetric encryption framework to enhance cryptographic security. In addition, a dynamic message topic mechanism was introduced under the MQTT communication protocol to improve data transmission security in IIoT environments. By integrating these components, an MQTT-based IIoT secure communication system with multiple robotic arms was successfully implemented and demonstrated.

Note that the proposed framework is currently designed and validated under a single-master, time-triggered polling-based industrial control architecture. It does not yet extend to multi-master configurations, fully asynchronous event-driven systems, or large-scale distributed IIoT networks. The extension of the proposed approach to these more complex and heterogeneous scenarios will be considered as future research directions.

Declaration of Conflicting Interests

The author(s) declare no potential conflicts of interest to the research, authorship, and/or publication of this article.

Funding

This work was funded by the National Science and Technology Council, Taiwan, under grant no. NSTC 115-2221-E-167-015-.

ORCID iD

Jun-Juh Yan https://orcid.org/0000-0002-9438-1937

References

1 R. Soua, M. R. Palattella, A. Stemper, and T. Engel: IEEE Internet Things J. **9** (2022) 14928. https://doi.org/10.1109/JIOT.2021.3129058
2 J. D. Rodríguez-Muñoz, E. Tlelo-Cuautle, and L. G. de la Fraga: Integr. **102** (2025) 102378. https://doi.org/10.1016/j.vlsi.2025.102378
3 K. Kaur, A. Kaur, Y. Gulzar, and V. Gandhi: Front. Comput. Sci. **6** (2024) 1420680. https://doi.org/10.3389/fcomp.2024.1420680
4 OASIS Open: OASIS Std. 3.1.**1** (2015). http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
5 N. Z. Wang and H. Y. Chien: IEEE Internet Things J. **11** (2024) 10768. https://doi.org/10.1109/JIOT.2023.3327447
6 R. Saha, G. Geetha, G. Kumar, and T. H. Kim: Secur. Commun. Netw. **2018** (2018) 9802475. https://doi.org/10.1155/2018/9802475
7 T. M. Fernández-Caramés and P. Fraga-Lamas: IEEE Access **8** (2020) 21091. https://doi.org/10.1109/ACCESS.2020.2968985
8 S. Gupta, K. Sau, J. Pramanick, S. Pyne, R. Ahamed, and R. Biswas: Proc. 8th Industrial Automation and Electromechanical Engineering Conf. (IEMECON) (2017) 162–167. https://doi.org/10.1109/IEMECON.2017.8079582
9 V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang: Int. J. Adv. Comput. Sci. Appl. **9** (2018) 405. https://doi.org/10.14569/IJACSA.2018.090354
10 A. Pathare and B. Deshmukh: Int. J. Mod. Trends Sci. Technol. **8** (2022) 141. https://doi.org/10.46501/IJMTST0801024
11 H. A. A. Mansour: J. Inf. Hiding Multimed. Signal Process. **12** (2021) 83.
12 Z. Yang, Y. Liu, Y. Wu, Y. Qi, F. Ren, and S. Li: Chaos Solitons Fractals **167** (2023) 113039. https://doi.org/10.1016/j.chaos.2022.113039
13 A. Kamadi and Z. Abbas: Microelectron. J. **123** (2022) 105410. https://doi.org/10.1016/j.mejo.2022.105410
14 B. Paul, G. Trivedi, P. Jan, and Z. Nemec: Proc. Int. Conf. Radioelektronika (2019). https://doi.org/10.1109/RADIOELEK.2019.8733468
15 H. Wen, J. Wu, L. Ma, Z. Liu, Y. Lin, L. Zhou, H. Jian, W. Lin, L. Liu, T. Zheng, and C. Zhang: IEEE Photonics J. **15** (2023) 1. https://doi.org/10.1109/JPHOT.2022.3233129
16 Q. Qin, Z. Liang, S. Liu, and C. Zhou: IEEE Access **11** (2023) 47873. https://doi.org/10.1109/ACCESS.2023.3267485
17 T. Nestor, A. Belazi, B. Abd-El-Atty, M. N. Aslam, C. Volos, N. J. De Dieu, and A. A. Abd El-Latif: Symmetry **14** (2022) 424. https://doi.org/10.3390/sym14020424
18 J. Liu, J. Zhang, and Y. Wang: IEEE Trans. Neural Netw. Learn. Syst. **35** (2024) 285. https://doi.org/10.1109/TNNLS.2022.3173516
19 O. Guillén-Fernández, E. Tlelo-Cuautle, L. G. de la Fraga, Y. Sandoval-Ibarra, and J. C. Núñez-Pérez: Mathematics **10** (2022) 1907. https://doi.org/10.3390/math10111907
20 C. Prajisha and A. R. Vasudevan: Int. J. Inf. Secur. **21** (2022) 1263. https://doi.org/10.1007/s10207-022-00611-9
21 A. Hue, G. Sharma, and J. M. Dricot: Electronics **11** (2022) 70. https://doi.org/10.3390/electronics11010070
22 C. Meshram, R. W. Ibrahim, A. J. Obaid, S. G. Meshram, A. Meshram, and A. M. Abd. El-Latif: J. Adv. Res. **32** (2021) 139. https://doi.org/10.1016/j.jare.2020.08.015
23 J. J. Yan, C. Y. Chen, and J. S. H. Tsai: Nonlinear Anal. Hybrid Syst. **22** (2016) 276. https://doi.org/10.1016/j.nahs.2016.05.004
24 G. S. Vicinansa and D. Liberzon: SIAM J. Control Optim. **61** (2023) 198. https://doi.org/10.1137/21M1411871
25 A. Rukhin, J. Soto, and J. Nechvatal: NIST Spec. Publ. 800-**22** (2010). http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf
26 B. H. Susanti, J. Jimmy, and M. W. Ardyani: Proc. Int. Semin. Res. Inf. Technol. Intell. Syst. (2021) 324. https://doi.org/10.1109/ISRITI54043.2021.9702862

About the Authors



Guo-Hsin Hu received his Ph.D. degree in electronic engineering from National Kaohsiung University of Science and Technology, Taiwan. His research interest focuses on smart manufacturing, text mining, and information technology. (guohsin@mail.mirdc.org.tw)



Wei-Zhe Xi received his B.S. degree from Shu-Te University of Technology, Taiwan in 2021. He is currently a master student in the Department of Electronic Engineering Science, National Chin-Yi University of Technology, Taichung, Taiwan. His main research interests are in the areas of chaotic systems, random number generator design, and IoT applications.



Jun-Juh Yan received his M.S. degree in electrical engineering from National Central University, Taiwan, in 1992 and his B.S. and Ph.D. degrees in electrical engineering from National Cheng Kung University, Taiwan, in 1987 and 1998, respectively. At present, he is a professor in the Department of Electronic Engineering, National Chin-Yi University of Technology, Taichung, Taiwan. His main research interests are in the areas of IoT applications, signal and image processing, chaos communication systems, variable-structure control systems, and adaptive control.