IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received July 1, 2020, accepted July 30, 2020, date of publication August 4, 2020, date of current version August 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3014217

Homomorphic Encryption of Supervisory Control Systems Using Automata

SIAN ZHOU[1], ZHENHUA YU[2], EMAD S. ABOUEL NASR[3,4], HAITHAM A. MAHMOUD[3,4], (Senior Member, IEEE), EMAD MAHROUS AWWAD[5], AND NAIQI WU[6], (Fellow, IEEE)

[1]Faculty of Information Technology, Macau University of Science and Technology, Taipa 999078, Macao
[2]Institute of Systems Security and Control, College of Computer Science and Technology, Xi'an University of Science and Technology, Xi'an 710054, China
[3]Industrial Engineering Department, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia
[4]Faculty of Engineering, Mechanical Engineering Department, Helwan University, Cairo 11732, Egypt
[5]Electrical Engineering Department, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia
[6]Institute of Systems Engineering and Collaborative Laboratory for Intelligent Science and Systems, Macau University of Science and Technology, Taipa 999078, Macao

Corresponding author: Zhenhua Yu (zhenhua_yu@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61873277, and the Key Research and Development Program of Shaanxi Province under Grant Nos. 2019GY–056 and 2018ZDXM–GY–036.

ABSTRACT Cyber-physical systems have been highly integrated into many contemporary infrastructures. As this integration deepens, the importance of protecting these systems from unauthorized access and data corruption increases. Nowadays, cyber-physical systems are not well protected against network attacks. One solution is to improve the security of a system by encrypting the transmitted data. In this paper, we consider the encryption of supervisors of discrete event systems modeled with deterministic finite-state automata. We propose an encryption framework of supervisory control systems based on the matrix notation of automata. The purpose of using matrix notation is to make it suitable for homomorphic encryption schemes over integers, which are emerging in the cryptography area. We calculate the entropy of the matrix notation and find that as the size of a system increases, it gets smaller and approaches zero. Owing to the low entropy of the matrix notation, we propose an algorithm to enhance its entropy. By applying the entropy-enhancing process, the distribution characteristics of entries in matrices or vectors can be hidden to avoid a brute force attack. Correspondingly, we propose an entropy restoration algorithm to ensure that the control action can be transmitted correctly.

INDEX TERMS Cyber-physical system, discrete event system, supervisory control, security, encryption, automaton.

I. INTRODUCTION

Cyber-physical systems (CPSs) are the integrations of computation, communication, control, and physical processes [1]. They realize the interaction of information flows between the physical world and the cyber-world. Today, individuals, society, industry, and every aspect of economic activities are highly dependent on this network system technology. Examples of CPSs include various complex human-made systems such as smart grids, robotics systems, oil and gas distribution systems, factory automation, and autonomous vehicle systems [2]–[5].

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiwu Li.

Intuitively, physical systems are considered to be able to effectively resist various attacks due to their closure and preset security mechanisms, thereby ensuring the security of CPSs. However, recent security incidents such as "Hide and Seek IoT Botnet" [6] and DTrack [7] have shown that, because of the interaction between physical and information systems through the network, cyber-attacks can directly affect the security of a CPS. The spread of data has become a fundamental feature of complex systems [8]. Notably, during the evolution of the traditional physical systems to CPSs, the operating environment of a system changes from being closed and isolated to open and interconnected. Therefore, in recent years, the security of CPSs has become a very fertile field of research [9].

From an engineering perspective, it is significant to model a CPS for design and development. First, in the day-to-day life of our technological and increasingly computer-dependent world, many processes are discontinuous. Secondly, in a typical CPS, embedded computers and networks usually monitor and control the physical processes through feedback loops where physical processes affect computations and vice versa [10]. Based on the above considerations, in this paper, we model a CPS as the supervisory control problem for discrete event systems. A discrete event system is a dynamic system whose state space is discrete and possibly infinite, and its dynamics is event-driven, not time-driven [11]. Supervisory control theory is designed for closed-loop control systems of discrete event systems and related security [12]. Fig. 1 depicts a typical supervisory control system.

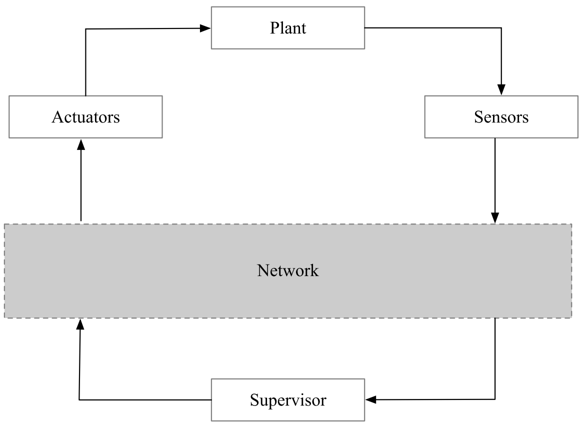

FIGURE 1. The controlled closed-loop system architecture.

According to the severity and involvement of cyber attacks, attacks can be divided into active and passive ones [13]. Passive attacks are those in which an attacker intercepts information flows between two parties to steal information stored in a system by eavesdropping or similar means. Opacity is an information flow property that characterizes the inability for an external intruder to know whether the given "secret" behavior happens in a system [14], [15]. However, almost all studies on opacity are based on data transmitted in plaintext data. A more general and straightforward way to hide information is to encrypt the transmitted data.

In the research of data encryption, a traditional method is mainly communication data encryption, that is, the data transmitted through the communication channel is encrypted and protected, thus significantly enhancing the security of the communication channel data.

However, the encryption of data by the conventional method is limited to the communication channel. Other data in the system, especially the critical data in the supervisor, is in a situation of lack of protection, which can bring serious security risks to the entire system.

When an attacker steals the plaintext data in a supervisor, he/she can use the data to break the cyber-physical system. First, the attacker can directly use essential data such as process signals and control policy to infer the current state of the plant. Second, the attacker can provide attack preparations for the implementation of subsequent covert attacks based on critical data in the supervisor, and then cause serious attack damage [16].

Given the shortcomings of communication encryption methods, in recent years, some researchers have proposed a new way to simultaneously encrypt the communication channel and the data in the supervisor, that is, the "encrypted supervisor" [17]–[21].

Better than communication encryption, the new method no longer needs to decrypt the input of the supervisor. It directly calculates the encrypted output from the encrypted input and control policy, thereby ensuring the security of the communication channel and the data in the supervisor. Even if the attacker successfully steals the data in the supervisor, because the data is encrypted and protected, as long as the data cannot be decrypted correctly, it is difficult to pose a threat and damage to the cyber-physical system. Specifically, the new encryption method is realized by using a homomorphic encryption scheme.

The notion of homomorphic encryption was first proposed by Rivest *et al.* in 1978 [22]. It is a form of encryption that allows computations on the ciphertext to produce an encrypted result that, after decryption, matches the result of the operation as if the operation is performed on the plaintext. Therefore, data can be kept confidential during processing, enabling data residing in untrusted environments to perform useful tasks.

After the first fully homomorphic encryption scheme is proposed [43], the research on homomorphic encryption schemes develops in a spurt. Due to the excellent property of homomorphic encryption schemes, they are widely used in many systems such as cloud systems [23], [24]. The first attempt to apply the homomorphic encryption scheme for a supervisor is in [17].

In [17], the authors encrypt a linear controller by using multiplicative homomorphic encryption schemes. Since the encryption schemes adopted do not apply to the addition operation in the controller, they extract part of the process of the controller to the outside, which reduces the coupling of the whole system. To overcome this difficulty, the study in [18] uses fully homomorphic encryption schemes to encrypt the controller.

In the majority of these works, a system to be considered is modeled as a continuous-variable dynamic system. The study in [21] is the first work to introduce homomorphic encryption schemes into controlled discrete event systems. In [21], the authors consider the design of a supervisor encryption scheme developed for networked control systems, where the supervisor is modeled as a signal interpreted Petri net [25]. However, the encryption scheme in [21] is symmetric, which means that although there is no decryption process inside the supervisor, secret keys need to be saved for the supervisor encryption.

Compared with the signal interpreted Petri nets, general Petri nets [26]–[36] and automata [37]–[39] are tools that are more basic and extensively applied for modeling and analysis. When using signal interpreted Petri nets to model a system, transitions are associated with a firing condition given as input signals, and places to specify output signals, which significantly affects its modeling power. Many researchers have used automata as models to study the network security of CPSs [39]. In this paper, we employ finite-state automata to model a system. To reduce the insecurity caused by the private key storage of a supervisor, we propose to use public-key homomorphic encryption schemes in the framework. The main contributions of this paper are stated as follows:

- A supervisor encryption framework is developed for supervisory control systems, where the supervisor is modeled as an automaton.
- A matrix notation of automata is proposed, which makes it possible to introduce encryption schemes into the encryption framework.
- Two algorithms for enhancing and restoring the min-entropy of matrices or vectors in matrix notation are developed to increase the security of the overall framework.

The remaining sections of this paper are organized as follows. Section II introduces the related preliminary concepts. Section III formally defines a novel matrix notation of automata. In Section IV, we calculate the entropy of the proposed matrix notations and propose two algorithms to increase and restore the entropy. Section V presents a supervisor encryption framework. Section VI selects an encryption scheme to illustrate the correctness of the framework. Finally, Section VII concludes the paper.

II. PRELIMINARIES

A. AUTOMATON MODEL

We consider plants modeled as deterministic finite-state automata. Such an automaton is denoted by $G = (Q, E, f, q_1)$, where Q is the finite set of states, E is the finite set of events, $f : Q \times E \to Q$ is the (potentially partial) transition function, and q_1 is the initial state. The state of an automaton at time epoch $k+1$ is denoted by $q(k+1)$ and is uniquely determined by its state $q(k)$ at the previous time epoch k and the event $e(k)$ applied to the system at time epoch k, that is:

$$q(k+1) = f(q(k), e(k)). \tag{1}$$

The term "time epoch k" is used to indicate the moment when the state of the automaton changes.

B. SUPERVISORY CONTROL THEORY

Supervisory control theory is a technology for automatically synthesizing supervisors that restrict the behavior of a system to satisfy given specifications [40]. The supervisor observes some, probably all, events executed by the plant. Then, the supervisor tells the plant which events in the current active event set of the plant are allowed next. More precisely, the supervisor can disable some, but not necessarily all, available events of the plant.

Whenever the supervisor observes the plant executing a new event, it can make the decision on which events to disable. If we use an automaton to represent the plant, then we can also use an automaton to describe the supervisor. We can use the parallel composition of automata as an algebraic means to capture the effects of a controlled closed-loop system [11].

Usually, due to the limited availability of sensors in real systems, only partial observations of states and events can be obtained directly [41]. However, this is an issue that should be considered when modeling a supervisor and is beyond the scope of the paper. Interested readers may refer to [11], [12], [31], [37].

C. HOMOMORPHIC ENCRYPTION

A homomorphic encryption scheme consists of the following four algorithms [42]: key generation, encryption, decryption, and evaluation. The key generation algorithm accepts a security parameter and outputs a public key pk and a secret key sk. The encryption algorithm accepts pk and a plaintext m, outputs the corresponding ciphertext c. The decryption algorithm takes sk and c, outputs the plaintext m. The evaluation algorithm accepts the public key pk, a function $\mathcal{F}$ and a set of ciphertexts $c_1, c_2, \ldots, c_n$, and outputs a ciphertext $c_{\mathcal{F}}$:

$$c_{\mathcal{F}} = Eval(pk, \mathcal{F}, c_1, c_2, \ldots, c_n). \tag{2}$$

The encryption scheme is correct for a function $\mathcal{F}$ if for any key-pair (pk, sk) generated by the key generation algorithm, any plaintexts $m_1, m_2, \ldots, m_n$ and the corresponding ciphertexts $c_1, c_2, \ldots, c_n$, the ciphertext $c_{\mathcal{F}}$ generated by the evaluation algorithm satisfies the following equation:

$$Dec(sk, c_{\mathcal{F}}) = \mathcal{F}(m_1, m_2, \ldots, m_n), \tag{3}$$

that is, the result calculated on the ciphertext after decryption is equal to the result calculated on the plaintext.

In the rest of the paper, we use $Enc(\cdot)$ and $Dec(\cdot)$ that do not specify the key used to denote encryption and decryption algorithms, respectively.

According to the scope of the function $\mathcal{F}$ supported on the ciphertext, homomorphic encryption schemes can be divided into partially homomorphic encryption schemes and fully homomorphic encryption schemes. If $\mathcal{F}$ can be an arbitrary function, the homomorphic encryption scheme is a fully homomorphic encryption scheme; otherwise, it is a partially homomorphic encryption scheme. Gentry [43] proposes the first fully homomorphic encryption scheme, but it cannot be put into practical application due to low calculation efficiency.

The functions $\mathcal{F}$ supported by current mainstream homomorphic encryption schemes include addition or multiplication or both.

Definition 1 (Additively Homomorphic Encryption): A homomorphic encryption scheme is additively homomorphic if an operation $\oplus$ exists such that:

$$Dec(c_1 \oplus c_2 \oplus \cdots \oplus c_n) = m_1 + m_2 + \cdots + m_n, \quad (4)$$

where $m_1, m_2, \ldots, m_n$ are plaintexts and $c_1, c_2, \ldots, c_n$ are the corresponding ciphertexts.

Definition 2 (Multiplicatively Homomorphic Encryption): A homomorphic encryption scheme is multiplicatively homomorphic if an operation $\otimes$ exists such that:

$$Dec(c_1 \otimes c_2 \otimes \cdots \otimes c_n) = m_1 \times m_2 \times \cdots \times m_n, \quad (5)$$

where $m_1, m_2, \ldots, m_n$ are plaintexts and $c_1, c_2, \ldots, c_n$ are the corresponding ciphertexts.

D. PUBLIC-KEY CRYPTOGRAPHY

Generally, there are mainly two cryptography categories: private- and public-key cryptography. In the setting of private-key encryption, two parties share a secret key and use this key when they want to communicate secretly. However, in contrast to private-key techniques, public-key cryptography enables parties to communicate privately without having agreed on any secret information in advance.

In a public-key setting, a party who wishes to communicate securely generates a pair of keys: a public key that is widely disseminated, and a private key that is kept secret [44]. Having generated these keys, a party can use them to ensure secrecy for messages that it receives using a public-key encryption scheme, or integrity for messages that it sends using a digital signature scheme [45]–[47].

The most apparent difference between private- and public-key encryption is that the former assumes complete secrecy of all cryptographic keys, whereas the latter requires secrecy for only the private key [44]. In other words, when we apply a public-key encryption scheme, as long as the private key is safe, the encrypted data can be trusted.

E. MIN-ENTROPY

Min-entropy is a measure of the difficulty for an attacker to guess the most likely secret in a system. In [48], the min-entropy of an independent discrete random variable X that takes values from the set $A = \{x_1, x_2, \ldots, x_P\}$ with probability $Pr[X = x_i] = p_i$ for $i \in \{1, 2, \ldots, P\}$ is defined as:

$$\begin{aligned} H_{min}(X) &= \min_{1 \le i \le P}(-log_2 p_i), \\ &= -log_2(\max_{1 \le i \le P}(p_i)). \end{aligned} \quad (6)$$

If X has min-entropy H, then the probability of observing any particular value of X is no greater than 2^{-H}. The maximum possible value of the min-entropy of a random variable with P different values is $log_2 P$, which is obtained when the random variable has a uniform probability distribution.

The lower the min-entropy of a random variable is, the more heterogeneous its distribution is. A low min-entropy indicates that there must be a subset of variables with a higher probability, and this subset can help an attacker to perform a group representation attack [49]. Therefore, high entropy is a necessary condition for security.

III. A NOVEL MATRIX NOTATION OF AUTOMATA

In [50], the authors propose a transition matrix notation of automata. They use different binary transition matrices to represent different events such that the structure of automata can be derived from the transition matrices. Here we propose a novel matrix notation of automata, using vectors to denote events, thereby separating the structure of automata from the mathematical representation of events.

Given an automaton $G = (Q, E, f, q_1)$, where $Q = \{q_1, q_2, \ldots, q_M\}$ is the finite set of states and $E = \{e_1, e_2, \ldots, e_N\}$ is the finite set of events. We use an $M \times 1$ binary indicator vector $q(k)$ to represent its state at time epoch k. More specifically, if the automaton is at state q_j and time epoch k, then $q(k)$ is a vector of zeros except a single nonzero entry with value "1" at the j-th position. We call $q(k)$ the state vector of the automaton. Similarly, we use an $N \times 1$ binary indicator vector $e(k)$ to represent the event occurs at time epoch k and call it the event vector of the automaton. Without causing ambiguity, we denote the i-th entry of the indicator vector $q(k)(e(k))$ as $q(k)_i(e(k)_i)$.

As shown in Eq. (1), the next state of an automaton is determined by the current state and the event that has occurred. Now we have vectors representing the current state and event, and only two vectors cannot describe this fact. In order to establish the relationship between the event vector and the state vector, and to achieve the purpose of characterizing the evolution of the automaton, we define an operation named *Events Trigger* to describe the automaton at state $q(k)$ and time epoch k and the event $e(k)$ that has occurred.

Definition 3 (Events Trigger): If event $e(k)$ happens and the current state is $q(k)$, we represent this situation by constructing a new vector as follows. Specifically, we multiply $e(k)$, which is a vector, by every entry of $q(k)$, which is a scalar, to obtain new vectors, and then arrange these new vectors to form a new vector $qe(k)$:

$$qe(k) = \begin{bmatrix} q(k)_1 e(k) \\ q(k)_2 e(k) \\ \vdots \\ q(k)_M e(k) \end{bmatrix}. \quad (7)$$

We call this operation *Events Trigger*, denoted as $qe(k) = ET(q(k), e(k))$.

Note that $qe(k)$ is a vector of size $M \times 1$ composed of vectors of size $N \times 1$. According to the definition of $q(k)$, there is only one entry with value "1" in $q(k)$, so $qe(k)$ is a vector of zero vectors except for a single nonzero entry with value $e(k)$ at the j-th position if the automaton is at state q_j and time epoch k, and $e(k)$ has occurred. Since $qe(k)$ is a vector in which each entry is a vector, it is not convenient for the computer to store and calculate. Therefore, we develop a process to solve this problem.

For convenience, we denote the i-th row of the matrix S as $row_i(S)$, the j-th column of the matrix S as $col_j(S)$, and the elements of the matrix S in the i-th row and j-th column as $S_{i,j}$.

Algorithm 1 Entry Scalarization Process

```
Input: $vM$, a vector or a matrix composed of vectors
Output: $vM^s$, a vector or a matrix composed of scalars
1: if $vM$ is a vector then
2:     $vM^s \leftarrow [\quad]$
3:     for $i = 1$ to $|vM|$ do
4:         for $j = 1$ to $|vM_i|$ do
5:             ADD $vM_{ij}$ to $vM^s$
6:         end for
7:     end for
8:     if $|vM^s_1| > 1$ then
9:         $vM \leftarrow vM^s$
10:        Go back to Step 2
11:    end if
12: else
13:    $vM^s \leftarrow [\quad]$
14:    for $i = 1$ to $|col_1(vM)|$ do
15:        for $j = 1$ to $|row_1(vM)|$ do
16:            for $k = 1$ to $|vM_{i,j}|$ do
17:                ADD $vM_{i,jk}$ to $row_i(vM^s)$
18:            end for
19:        end for
20:    end for
21:    if $|vM^s_{1,1}| > 1$ then
22:        $vM \leftarrow vM^s$
23:        Go back to Step 13
24:    end if
25: end if
26: return $v^s_M$
```

After executing the *Entry Scalarization* process, we get a new vector without changing the order relationship within and between each entry, just like breaking the boundaries between different entries. For convenience, we use $ES(\cdot)$ to denote Algorithm 1.

Example 1: Given a vector $v = [[1, 0], [0, 1]]$, which is a vector composed of two vectors $[1, 0]$ and $[0, 1]$, usually we need two address spaces in the computer to store them. After executing Algorithm 1, we have $v^s = [1, 0, 0, 1]$, which is a vector composed of scalars and only needs one address space. $\diamondsuit$

In the same way, we can find $qe^s(k) = ES(qe(k))$ as a vector of size $MN \times 1$.

We know that any automaton can be shown graphically by a state transition diagram where the states are represented by the vertices and each edge represents the event that causes the state transition. In other words, the states of an automaton are all connected by events. Thus, we can represent the structure of an automaton by recording events between each state.

To represent this kind of connection, we use a binary indicator vector of size $1 \times N$ and denote it as $e_{ij}(i, j \in M)$. If there are several events $e_{k_1}, e_{k_2}, \cdots, e_{k_m}$ from state q_i to q_j, then e_{ij} is a vector of zeros except nonzero entries with value "1" at the k_t-th position ($t \in \{1, \cdots, m\}$). We call it the connection vector between states.

Then, we construct a structural matrix S of size $M \times M$ to denote the automaton's structure. Each entry at the (j-th, i-th) position of the structural matrix is the connection vector e_{ij}, denoting events from q_i to q_j. For the sake of calculation, we apply Algorithm 1 to S to obtain a new structural matrix S^s of size $M \times MN$. Thus, every N column represents all connection relationships between different states.

With the above notations, we have

$$\begin{aligned} q(k+1) &= S \times ET(q(k), e(k)) \\ &= ES(S) \times ES(ET(q(k), e(k))). \end{aligned} \tag{8}$$

Theorem 1: Eq. (8) accurately describes the transition function of an automaton, that is, Eq. (8) is the equivalent representation of Eq. (1) in the matrix notation of automaton.

Proof: We consider an automaton with M states and N events, and assume that at time epoch k, the automaton is at state q_i, and event e_t occurs. The next state of the automaton becomes q_j. As expressed in Eq. (1), it is:

$$q_j = f(q_i, e_t). \tag{9}$$

Now we represent q_i and e_t with the binary indicator vectors mentioned earlier, and then we need to verify that after the operation of Eq. (8), the resulting vector is a representation of q_j. By definition, we have:

$$\begin{aligned} q(k) &= [\overbrace{0, \cdots, \underset{\substack{\uparrow \\ i\text{-th}}}{1}, \cdots, 0}^{M}]^T, \\ e(k) &= [\overbrace{0, \cdots, \underset{\substack{\uparrow \\ t\text{-th}}}{1}, \cdots, 0}^{N}]^T. \end{aligned} \tag{10}$$

$$S = \begin{bmatrix} \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & [\cdots, \underset{\substack{\uparrow \\ t\text{-th}}}{1}, \cdots] & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots \end{bmatrix}. \tag{11}$$

As shown in Eq. (11), in the structural matrix of the automaton, we only indicate the t-th entry of the (j-th, i-th) connection vector since the other entries do not affect the calculation of Eq. (8). After the operation *Events Trigger*, we have:

$$qe(k) = \begin{bmatrix} [0, \cdots, 0]^T \\ \vdots \\ e(k) \\ \vdots \\ [0, \cdots, 0]^T \end{bmatrix} \leftarrow i\text{-th} . \tag{12}$$

Note that S is an $M \times M$ matrix, where each entry is a $1 \times N$ vector; $qe(k)$ is an $M \times 1$ vector, where each entry is an $N \times 1$

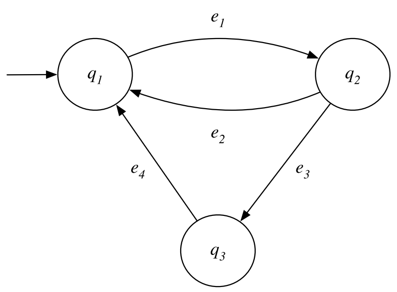

FIGURE 2. Automaton in Examples 2 and 3.

vector. According to the definition of matrix multiplication, we can multiply S and $qe(k)$ and obtain the following result:

$$q(k+1) = [\overbrace{0, \cdots, \underset{\underset{j\text{-th}}{\uparrow}}{1}, \cdots, 0}^{M}]^T, \tag{13}$$

which is a state vector that represents q_j and conforms to Eq. (9).

The *Entry Scalarization* process on S and $qe(k)$ is only for the convenience of computer processing and simplified display, and will not affect the calculation results. □

Example 2: Consider the automaton in Fig. 3. It has three states and four events, therefore we can define the following structural matrix:

$$S^s = \begin{bmatrix} 0000 & 0100 & 0001 \\ 1000 & 0000 & 0000 \\ 0000 & 0010 & 0000 \end{bmatrix}. \tag{14}$$

For ease of understanding, we write every four column in a group. For example, the first four columns of S^s indicate that there is only one event e_1 from q_1 to q_2, and there is no transition to q_3 or self-loop to q_1. Assume that at the time epoch k, the state of the automaton is q_2, and event e_3 occurs. Then we have:

$$q(k) = [0, 1, 0]^T, \quad e(k) = [0, 0, 1, 0]^T,$$
$$ES(ET(q(k), e(k))) = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]^T.$$

From Eq. (8), we have $q(k+1) = [0, 0, 1]^T$, which means that the next state of the automaton is q_3. This result matches the state transitions of the automaton. ♢

In this paper, we model the supervisor as an automaton, and the control action is reflected from the active event set of the current state of the supervisor. Thus we need to obtain the active event set from the structural matrix and the current state vector, which can be achieved by Algorithm 2.

In Algorithm 2, *Esize* in Line 2 represents the number of events in an automaton, which is equivalent to $Esize = \frac{MN}{M} = N$. Lines 3–7 determine the current state of the automaton from the state vector. The current state of the automaton is $q_{currentLo}$. Lines 9–17 determine the current active event set. Every of *Esize* columns in S^s corresponds to the related column of the connection vector in S. The time complexity of Algorithm 2 is $\mathcal{O}(MN)$, where M is the number of states of an automaton and N is the number of events.

Algorithm 2 Active Event Set Generator

```
Input: $q(k)$, the current state vector; $S^s$, the structural matrix
    of the automaton
Output: $AEset$, the active event set of the current state
 1: $AEset \leftarrow \emptyset$
 2: $Esize \leftarrow |row_1(S^s)| \;/\; |q(k)|$
 3: for $i = 1$ to $|q(k)|$ do
 4:     if $q(k)_i = 1$ then
 5:         $currentLo \leftarrow i$
 6:     end if
 7: end for
 8: $J \leftarrow Esize \times (currentLo - 1) + 1$
 9: for $j = J$ to $(J + Esize - 1)$ do
10:     for $k = 1$ to $|col_1(S^s)|$ do
11:         if $S^s_{k,j} == 1$ then
12:             if $e_{j-J+1}$ not in $AEset$ then
13:                 ADD $e_{j-J+1}$ to $AEset$
14:             end if
15:         end if
16:     end for
17: end for
18: return $AEset$
```

Example 3: Consider the automaton in Fig. 2, and its structural matrix is shown in Eq. (14). Assume that the current state vector is $q(k) = [0, 1, 0]^T$. By running Algorithm 2, we can first obtain the number of events of the automaton from Line 2, i.e., $Esize = 4$. Then from Lines 3–7, we can determine the current state as q_2, and finally we can find that the current active event set is $\{e_2, e_3\}$ through Lines 9–17. Obviously, the running result of Algorithm 2 is in line with the facts. ♢

IV. EVALUATION AND IMPROVEMENT OF MATRIX NOTATION

Before applying the encryption algorithm, we need to calculate the entropy of the matrix notation of an automaton. If the entropy of this notation is low, it needs to be improved accordingly, since in practice, low-entropy data can be easily cracked by brute force.

A. MIN-ENTROPY OF ONE ENCRYPTION

In the above matrix notation, all entries of a matrix or vector are binary, i.e., 0 and 1. Even if they are evenly distributed, i.e., their probability is 0.5 in the plaintext space, according to Eq. (6), we can find that the min-entropy of the plaintext is only one, which is far lower than the requirements for practical applications. Here we give more accurate calculations and find that the facts are worse than what we expect.

From Section III, we can quickly know that to represent the structure of an automaton, in the matrix notation, the number of entries "0" is much larger than that of entries "1". Now we consider the worst case, that is, increase the number of entries "0" as much as possible, for the reason that the higher the probability of the most probable element in the plaintext space is, the lower the min-entropy of the plaintext space is.

Therefore, an automaton with N states and $N-1$ events is constructed (no isolated state because it does not make sense). The structural matrix of the automaton has (N^3-N^2) entries with $(N-1)$ entries "1" and (N^3-N^2-N+1) entries "0". The state vector has only one entry "1" and $(N-1)$ entries "0". Similarly, the event vector has one entry "1" and $(N-2)$ entries "0".

If we put all the entries of the structural matrix, the state vector, and the event vector together as the plaintext data, the min-entropy of the plaintext is $-log_2(\frac{N^3-N^2+N-2}{N^3-N^2+2N-1})$. It is not difficult to find that as N increases, the min-entropy of the plaintext approaches to 0.

B. WAYS TO INCREASE AND RESTORE THE ENTROPY

If the entropy of the plaintext is too low, we use either an encryption scheme for low-entropy data or perform an entropy increase operation before encryption, without affecting the correctness of the decryption.

Before proposing a method to increase entropy, we analyze the operation process of the matrix notation first. By analyzing Eq. (8), the core equation representing the state transition of the automaton, we find that the right side of the equation includes two operations and one process: *Event Trigger*, *Entry Scalarization*, and matrix multiplication.

Here we call the operation between entries a meta-operation and analyze the types of meta-operations. Consider the *Event Trigger* first. Since both the state vector and event vector have only one entry "1", there is only one "1×1" meta-operation in the entire operation, and the other operations are "0×0" meta-operations or "0×1" meta-operations. The *Entry Scalarization* process does not increase the type of meta-operations. Matrix multiplication adds two kinds "$0+0$" and "$0+1$" meta-operations. Through the above analysis, we know that there are five kinds of meta-operations in the entire calculation process:

$$0+0=0, \quad 0+1=1, \ 0\times 0=0, \ 0\times 1=0, \ 1\times 1=1,$$

which is similar to odd and even operations:

$$\begin{aligned}&even+even=even, even+odd=odd,\\&even\times even=even, \ even\times odd=even, \ odd\times odd=odd.\end{aligned}$$

Therefore, we can increase the entropy of the matrix notation by using random large odd numbers to represent entries "1" and random large even numbers to represent entries "0". The following two algorithms are used to increase entropy and recover entropy, respectively. Here we use $s \leftarrow_\$ S$ to denote a value s chosen uniformly and randomly from the discrete set S.

Here, ρ is a security parameter, which is the degree of security that we want to achieve. We can see that after Algorithm 3 is adopted, all entries in the matrix or vector are evenly distributed in 2^ρ numbers, and their entropy is ρ. In practice, with $\rho \geq 32$, we can provide sufficient entropy

Algorithm 3 Entropy-Enhancing Process

```
Input: weakEntry, $\rho$
Output: strongEntry
1: if weakEntry $= 0$ then
2:     strongEntry $\leftarrow_\$ [1, 2^{\rho+1}]$ and strongEntry is even
3: else
4:     strongEntry $\leftarrow_\$ [1, 2^{\rho+1}]$ and strongEntry is odd
5: end if
6: return strongEntry
```

for the desired security, since breaking the system would require more than a billion of guesses [24].

Although Algorithm 3 enhances entries "0" and "1" to big random numbers, the parity of the enhanced entries unchanged, thereby Algorithm 4 only needs to perform inverse deduction based on the parity of the enhanced data.

Algorithm 4 Entropy Restoration Process

```
Input: strongEntry
Output: weakEntry
1: if strongEntry is even then
2:     weakEntry $\leftarrow 0$
3: else
4:     weakEntry $\leftarrow 1$
5: end if
6: return weakEntry
```

Remark 1: Note that in Algorithm 3, we set the upper limit of the random number interval to $2^{\rho+1}$ instead of 2^ρ, since we specify its parity when randomly selecting entries, which means that they are randomly distributed over 2^ρ numbers instead of $2^{\rho+1}$ numbers. As a qualified encryption algorithm does not preserve the parity of the plaintext, we do not have to worry about whether the above algorithms will bring insecurity.

It is not difficult to find that the time complexity of Algorithms 3 and 4 are both $\mathcal{O}(1)$. Since they are applied to each entry in the vector or matrix, when we enhance or restore the entropy of the state vector, the time complexity of the operation is $\mathcal{O}(M)$, where M is the number of states of the automaton. Similarly, the time complexity of applying Algorithms 3 and 4 to an event vector is $\mathcal{O}(N)$, where N is the number of events of the automaton. The time complexity of applying Algorithms 3 and 4 to a structural matrix is $\mathcal{O}(M^2N)$, where M is the number of states of the automaton, and N is the number of events of the automaton.

Example 4: Consider the structural matrix in Example 2. If we set $\rho = 7$, the matrix after entropy-enhancing is $[S^s]_e$, as shown at the bottom of the next page.

There are 36 entries in the original structural matrix, and the number of entries "0" is dominant. If we treat these 36 entries as a plaintext space, the probability of occurrence of "0" is 0.889. According to Eq. (6), the min-entropy of the original structural matrix is 0.169.

After performing Algorithm 3 on the original structural matrix, we can see that all its entries are evenly distributed over the 2^7 different integers. That is, the probability of all entries occurring is 2^{-7}, which means that the min-entropy of the new structural matrix is seven and is significantly improved relative to 0.169. $\Diamond$

V. SUPERVISOR ENCRYPTION FRAMEWORK

The basic idea of the Supervisor Encryption Framework is to use a public-key homomorphic encryption scheme such that there is no private key in the supervisor, which reduces the distribution of keys in the entire framework and enhances security. We add two interfaces to the original closed-loop system: an encryptor and a decryptor. As shown in Fig. 3, the encryptor is an interface between the sensor and the supervisor, and the decryptor is an interface between the supervisor and the actuator. The public key is stored in the encryptor and supervisor, which is used to encrypt the data obtained from the sensor and the structural matrix of the supervisor. The private key is stored in the decryptor, which is used to decrypt the control action obtained from the supervisor.

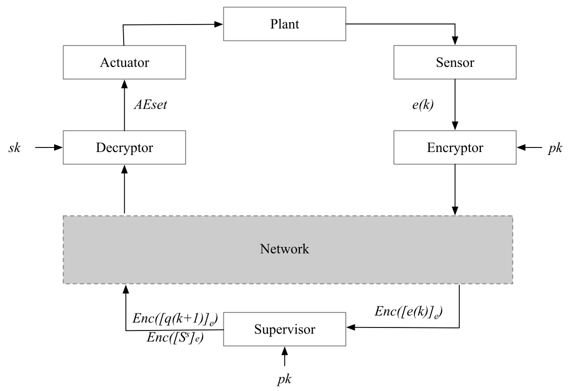

FIGURE 3. **The basic encryption framework.**

A. FEASIBILITY OF APPLYING HOMOMORPHIC ENCRYPTION SCHEME

In this paper, we consider the case where both the plant and the supervisor are modeled as automata. When we use an automaton to model the supervisor, we can use the set active events at the current supervisor's state as the control action, which can be obtained from Algorithm 2. Therefore, we can say that the current state of the supervisor represents the control action, and the state transition of the supervisor represents the control policy. In other words, when the supervisor is modeled as an automaton, we can use Eq. (1) to represent the supervisor. We use the symbol $\mathscr{S}_A$ to represent the supervisor modeled as an automaton. Based on the above information, we can decide whether a supervisor can be homomorphically encrypted.

Definition 4 (Encrypted Supervisor): Given a supervisor $\mathscr{S}_A$ that satisfies Eq. (1) and a homomorphic encryption scheme $\{KGen, Enc, Dec, Eval\}$, if there exists a map f_ε such that:

$$f_\varepsilon(Enc(q(k)), Enc(e(k))) = Enc(f(q(k), e(k))), \tag{15}$$

we say that the given $\mathscr{S}_A$ can be homomorphically encrypted through a specified encryption scheme. We call f_ε as an encrypted control policy.

When we apply the matrix notation to $\mathscr{S}_A$, we can use a multiplicatively and additively homomorphic encryption scheme to achieve this goal. Here, we apply a scalar encryption scheme, that is, when we talk about encrypting a matrix or a vector, we encrypt each entry of them separately.

According to the nature of multiplicatively and additively homomorphic encryption schemes, we have

$$\begin{aligned} Enc(m_1) \otimes Enc(m_2) &= Enc(m_1 \times m_2), \\ Enc(m_1) \oplus Enc(m_2) &= Enc(m_1 + m_2), \end{aligned} \tag{16}$$

where $\oplus$ and $\otimes$ are specific operations on the ciphertext space, and m_1 and m_2 are arbitrary plaintexts. We can think of them as multiplication and addition operations in the ciphertext space. By these two operations, we can sequentially define scalar-vector multiplication and matrix multiplication in the ciphertext space.

Definition 5 (Scalar-Vector Multiplication in the Ciphertext Space): Given an encrypted scalar c and an encrypted vector vc, the product of c and vc is defined as an vector C with entries

$$C_i = c \otimes vc_i, \tag{17}$$

and we denote the scalar-vector multiplication on the ciphertext space as $C = c \otimes vc$.

Definition 6 (Matrix Multiplication in the Ciphertext Space): Given two encrypted matrices EA and EB, if the number of columns in EA, with entries ea_{ij}, and the number of rows in EB, with entries eb_{ij}, are equal, then the product of EA and EB is defined as a matrix EC with entries

$$ec_{ij} = ea_{i1} \otimes eb_{1j} \oplus ea_{i2} \otimes eb_{2j} \oplus \cdots \oplus ea_{in} \otimes eb_{nj}, \tag{18}$$

where n is the number of columns in EA. To avoid confusion, we denote the matrix multiplication on the ciphertext space as $EC = EA \circledast EB$.

Obviously, the scalar-vector multiplication is homomorphic under multiplicatively homomorphic encryption scheme; matrix multiplication is homomorphic under

$$[S^s]_e = \begin{bmatrix} 240 & 30 & 206 & 118 & 220 & 135 & 238 & 118 & 84 & 88 & 190 & 155 \\ 229 & 230 & 162 & 54 & 108 & 192 & 236 & 178 & 190 & 170 & 200 & 236 \\ 168 & 216 & 12 & 218 & 146 & 38 & 63 & 234 & 232 & 164 & 204 & 202 \end{bmatrix}$$

multiplicatively and additively homomorphic encryption scheme, i.e.,

$$Enc(m) \otimes Enc(vm) = Enc(m \times vm),$$
$$Enc(A) \circledast Enc(B) = Enc(A \times B), \quad (19)$$

where m is a scalar in the plaintext space, vm is a vector in the plaintext space, and A and B are two arbitrary matrices that satisfy matrix multiplication condition in the plaintext space.

Theorem 1 makes sense since it highlights the equivalence of Eqs. (8) and (1). Therefore, we can use the matrix notation of an automaton to illustrate the feasibility of using a homomorphic encryption scheme to construct an encryption supervisor. We first establish two lemmas to illustrate the homomorphism of the operation *Events Trigger* and the *Entry Scalarization* process under homomorphic encryption scheme.

Lemma 1 The operation *Events Trigger* can maintain homomorphic properties under multiplicatively homomorphic encryption scheme. That is, there is a mapping f_{ET} in the ciphertext space that makes the following equation true:

$$f_{ET}(Enc(q(k)), Enc(e(k))) = Enc(ET(q(k), e(k)). \quad (20)$$

Proof: To prove this lemma, we construct the f_{ET} as:

$$f_{ET}(Enc(q(k)), Enc(e(k))) = \begin{bmatrix} Enc(q(k))_1 \otimes Enc(e(k)) \\ Enc(q(k))_2 \otimes Enc(e(k)) \\ \vdots \\ Enc(q(k))_M \otimes Enc(e(k)) \end{bmatrix}. \quad (21)$$

According to the property of multiplicatively homomorphic encryption schemes and Definition 5, we can readily deduce that Eq. (16) holds. □

Lemma 2: The *Entry Scalarization* process can maintain homomorphic properties under homomorphic encryption scheme.

Proof: Since the plaintext data that we encrypt here is a single entry in matrices or vectors, and the *Entry Scalarization* process does not change the relative position between entries or the value of the entry, the operation does not affect the encryption result, which means that the following equation holds:

$$ES(Enc(m)) = Enc(ES(m)), \quad (22)$$

where m is a matrix or vector. □

Theorem 2: In a discrete event system, any supervisor modeled as a finite-state automaton can be homomorphically encrypted by appropriately choosing an additively and multiplicatively homomorphic encryption scheme.

Proof: As shown in Section III, when a supervisor is modeled as a deterministic finite-state automaton, we can use the matrix notation to represent a supervisor. From Theorem 1, we can use Eq. (8) to characterize the operation of the supervisor.

Therefore, furthermore, with Definition 4, when we use the matrix notation to represent a supervisor, if the supervisor is homomorphically encrypted, there exists a map f_ε that satisfies the following equation:

$$f_\varepsilon(Enc(q(k)), Enc(e(k)), Enc(S^s)) = Enc(S^s \times ES(ET(q(k), e(k)))). \quad (23)$$

We construct f_ε as:

$$f_\varepsilon(Enc(q(k)), Enc(e(k)), Enc(S^s)) = Enc(S^s) \circledast ES(f_{ET}(Enc(q(k), Enc(e(k))))), \quad (24)$$

where f_{ET} satisfies Eq. (20).

From Lemmas 1 and 2, and Eq. (19), we can verify that Eq. (23) holds:

$$\begin{aligned} &f_\varepsilon(Enc(q(k)), Enc(e(k)), Enc(S^s)) \\ &= Enc(S^s) \circledast ES(f_{ET}(Enc(q(k), Enc(e(k))))) \\ &= Enc(S^s) \circledast ES(Enc(ET(q(k), e(k)))) \\ &= Enc(S^s) \circledast Enc(ES(ET(q(k), e(k)))) \\ &= Enc(S^s \times ES(ET(q(k), e(k)))), \end{aligned} \quad (25)$$

which means that f_ε is the encrypted policy.

In summary, the theorem is proved. □

B. INFORMATION FLOW IN THE FRAMEWORK

The data circulating in the entire framework can be seen in Fig. 3. Due to the low entropy nature of the matrix notation, we need to increase the entropy before encrypting the plaintext data and restore it after decryption, as shown in Fig. 4. For brevity, we call it compound encryption and compound decryption operations. Here, we add a pair of brackets and a subscript e to the data to indicate that the data is entropy-enhanced.

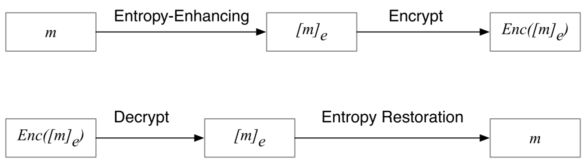

FIGURE 4. Compound encryption and compound decryption operations.

For the sake of easy understanding, we divide the operation of the entire encryption framework into two stages: initialization and normal operation.

In the initialization stage, the supervisor formulates the corresponding control policy according to the structure of the plant and obtains the structural matrix S and the initial state vector q_1. Then, we perform *Entry Scalarization* process and compound encryption operation on S to get $Enc([S^s]_e)$; and perform compound encryption operation on q_1 to obtain $Enc([q_1]_e)$. Finally, the supervisor sends $Enc([S^s]_e)$ and $Enc([q_1]_e)$ to the decryptor, and the decryptor can obtain the initial control action through decryption, entropy restoration, and Algorithm 2.

After entering the normal operation stage, the sensor monitors the corresponding data from the plant, which is $e(k)$, and sends it to the encryptor. The encryptor performs the compound encryption operation on $e(k)$ to obtain $Enc([e(k)]_e)$, and sends it to the supervisor. Then the supervisor can obtain the encrypted next state $Enc([q(k+1)]_e)$ through Eq. (24). Finally, the supervisor sends $Enc([S^s]_e)$ and $Enc([q(k+1)]_e)$ to the decryptor, and the decryptor can obtain the current control action through the same operation as in the initialization stage.

Remark 2: The underlying encryption framework proposed in this work is based on a fully homomorphic encryption scheme that can perform arbitrary operations on encrypted data. However, for higher efficiency, a somewhat homomorphic encryption scheme can also be used. Somewhat homomorphic encryption schemes are homomorphic encryption schemes that support addition and multiplication, but are limited to evaluating low–degree polynomials over encrypted data.

If we use a somewhat homomorphic encryption scheme, repeated operations on the ciphertext may increase the noise generated during the encryption process, resulting in decryption failure. To solve this problem, we can use the method in [21] to add an internal channel between the decryptor and the encryptor to prevent the increase of noise by refreshing the ciphertext of the state vector. As shown in Fig. 5, after one encryption iteration, the decryptor sends $q(k+1)$ to the encryptor. The encryptor updates $q(k)$ to the data received from the decryptor, and encrypts it in the next round of encryption operation, and then sends it to the supervisor.

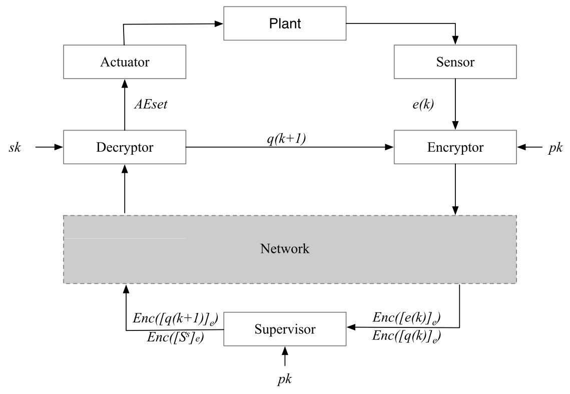

FIGURE 5. Adjusted encryption framework.

C. SECURITY ANALYSIS

The structural matrix S^s and signals $e(k)$ are processed within the supervisor to yield the control action, $AEset$. If someone performs unauthorized accesses to the supervisor, then S^s and $e(k)$ could be monitored and be used to infer the operating status of the plant. From an engineering perspective, it is essential to discuss protection methods that prevent malicious users from recording and stealing parameters and signals. In this framework, all the information appearing in the supervisor is encrypted, and the encrypted output is calculated directly from the encrypted input using the encrypted automata model without any decryption process inside. The final supervisor runs in an encrypted environment, which may help to enhance the network security of the cyber-physical system.

This framework is used to resist passive attacks. It is assumed that an intruder has all the knowledge of the system but does not know the secret key. All data transmitted over the network is encrypted ciphertext. Even if the intruder obtains the public key, based on the nature of the public-key encryption scheme, the intruder still knows nothing about the data transmitted in the network. The security of the framework is based on the safety of the selected encryption scheme. As long as the selected scheme is not broken, the whole closed-loop system is secure.

VI. A PRACTICAL EXAMPLE

To illustrate the implementation of the supervisor encryption framework described in Section V, let us consider the material handling system shown in Fig. 6, which consists of two AGV (automated guided vehicles) that move on two different tracks. One AGV serves three stations (A, B, C); the other serves two stations (C and D). We assume that the initial and final position of the first AGV is in station A, and the initial and final position of the second AGV is in station D. We assume that event e_2 is uncontrollable. The desired behavior of the material handling system is that two AGVs should not be present at station C at the same time to avoid a collision.

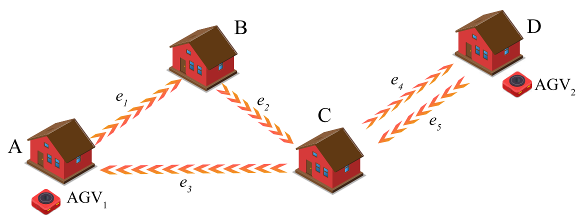

FIGURE 6. A material handling system.

The plant model is shown in Fig. 7 and the corresponding supervisor model is shown in Fig. 8. State XX' means that AGV$_1$ is in station X, and AGV$_2$ is in station X'. $(X \in \{A, B, C, \}, X' \in \{C, D\})$

A. SELECTED SCHEME–HE^L

First, we need to choose a suitable encryption algorithm. It should be noted that any multiplicatively and additively homomorphic encryption scheme over integers with public-keys can be used for this framework. Here we choose the HE^L encryption scheme in [51]. Its security is ensured due to the hard problem of solving the two-element Partial

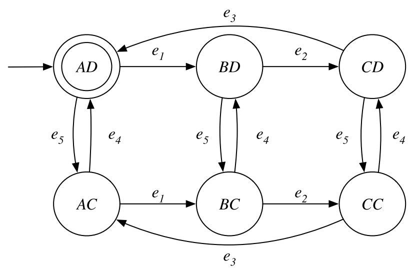

FIGURE 7. The plant model of the material handling system.

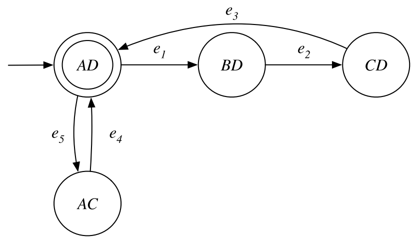

FIGURE 8. A supervisor of the material handling system.

Approximate Greatest Common Divisor. It consists of the following four algorithms:

- **Key generation** Choose two random odd integers P and R of size e and e', respectively. Choose two g − bit random integers Q_0, Q_1, i.e., $Q_i \leftarrow_\$ \mathbb{Z} \cap [0, 2^g/P)$, for $i = 0, 1$. Take a random integer $R' \leftarrow_\$ \mathbb{Z} \cap [2^{r'-1}, 2^{r'})$. Compute $X_0 = PQ_0$, $X_1 = PQ_1 + RR'$. Output the secret key, $SK = (P, R)$ and the public key, $PK = (X_0, X_1)$.
- **Encryption** Choose two s-bit random integers N_1, N_2, such that $N_2 > N_1$, and N_2 is an even number. Compute $X_2 = (N_1X_1) \bmod X_0$. The ciphertext $C = (M + N_2X_2) \bmod X_0$.
- **Decryption** We can obtain the plaintext from the ciphertext c and the equation $m = (C \bmod P) \bmod R$.
- **Evaluation** The addition and multiplication of two ciphertexts are given by $Add(C, C') = C + C' (\bmod X_0)$ and $Mult(C, C') = CC' (\bmod X_0)$.

Assume that the size of the plaintext integer to be encrypted is p, which may be taken as $\mathcal{O}(n)$. The suggested theoretical parameter setting is: $e = \mathcal{O}(n^3)$, $e' = \mathcal{O}(n^2)$, $g = \mathcal{O}(n^4)$, $s = n$, and $r' = n$.

B. A LIGHTWEIGHT APPLICATION

To illustrate the proposed framework, we set the security parameters ρ here to be 2, then we have $p = 4$, $n = 4$, $e = 64$, $e' = 16$, $s = 4$, $r' = 4$, and $g = 256$.

We name states AD, BD, CD, AC as q_1, q_2, q_3, q_4, respectively. The structural matrix of the supervisor is:

$$S^s = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}.$$

By applying Algorithm 3, we have $[S^s]_e$, as shown at the bottom of the page.

The matrix $[S^s]_e$ after encryption $Enc([S^s]_e)$ is dropped here due to the space.

Suppose that we have completed the initialization of the supervisor, that is, AGV$_1$ is serving at station A, AGV$_2$ is serving at station D, and the events allowed at this time k are e_1 and e_5. The plant model and the supervisor are in state AD, which means $q(k) = [1, 0, 0, 0]^T$.

Suppose that e_5 occurs at time epoch k, then the data sent by the sensor to the encryptor is $e(k) = [0, 0, 0, 0, 1]^T$.

After receiving $e(k)$, the encryptor performs a compound encryption operation on $e(k)$ to obtain $Enc([e(k)]_e) = [$ 220973000295354389689532135713938992106046547512144607039947513791871892537 61, 160707636578439556137841553246501085168033852736105168756325464575906830936 56, 200884545723049445172301941558126356460042315920131460945406830719883538670 56, 232388790146648307989485465237878996414518694942155322437434591967776428927 72, 532122401500999854721976410151199588406756190009359583207451508879380547107 96$]^T$ and sends it to the supervisor.

After receiving $Enc([e(k)]_e)$, the supervisor first encrypts $q(k)$ to obtain $Enc([q(k)]_e) = [$ 268229366930752683915307421233567952037761116045180399277989155663711227923 32, 261149909439964278723992524025564263398055010696170899229028879935848600271 73, 120530727433829667103381164934875813876025389552078876567244098431930123202 44, 380087429643571324801411721536571134576254421435256845148179649199557533648 51$]^T$, and then operates according to Eq. (24) to obtain $Enc([q(k+1)]_e) = [$ 703786633478338404940069554107130586350167009759667401927840605716784751714192 7, 584248345763163259443509344254622072908976423438110911705292339075718102481993 3, 668732678633069022774618717499363814259570137081548983609299932443264781508400 7,

$$[S^s]_e = \begin{bmatrix} 8&6&12&14&4&12&6&6&2&12&14&16&7&14&10&12&4&4&9&8 \\ 7&6&14&6&10&4&6&2&12&8&10&16&6&2&2&8&6&6&12&12 \\ 10&16&8&12&4&14&9&12&2&2&14&4&2&6&8&10&8&14&4&10 \\ 16&8&10&14&13&14&12&4&2&4&6&14&14&4&14&2&12&16&12&16 \end{bmatrix}$$

7229955363066952836782980037081371604670097983067 67770172881921461598356313759 3]^T.

Finally, the supervisor sends $Enc([S^s]_e)$ and $Enc([q(k+1)]_e)$ to the decryptor. After compound decryption, we have:

$$q(k+1) = [0, 0, 0, 1]^T$$

Following Algorithm 2, the active event set of $q(k+1)$ is $\{e_4\}$, that is, the control action after the supervisor observes e_5 is that e_4 is allowed to occur.

C. SIMULATION STUDY

Following the steps in Section V, we simulate the behavior of the material handling system under control with different security parameters to compare the performance and computation time to finish the whole encryption-decryption processes. The encryption shceme is HE^L mentioned above, and parameters are set based on ρ: $n = \rho + 2$, $e = n^3$, $e' = n^2$, $g = n^4$, $s = n$, and $r' = n$, where ρ is the min-entropy of the matrix notation after it is enhanced. The computation is done with Python programming language on macOS Catalina over a laptop with Intel(R) i7-8750H CPU at 2.20GHz and 16GB of RAM. Due to limited computing power, only four values of ρ are considered.

Table 1 shows the time that is taken for the supervisor to run at different stages, and Table 2 shows the time that is taken for the encryptor and decryptor to run.

TABLE 1. Time consumption of supervisor w.r.t. ρ.

ρ	Initialization Stage(s)	Normal operation Stage(s)
2	6×10^{-4}	1.5×10^{-4}
16	4.3×10^{-3}	1.7
24	1.7×10^{-2}	30.4
32	7.1×10^{-2}	297.6

TABLE 2. Time consumption of encryptor and decryptor w.r.t. ρ.

ρ	Encryptor(s)	Decryptor(s)
2	3.5×10^{-5}	7.1×10^{-5}
16	2.6×10^{-4}	6.9×10^{-2}
24	9.7×10^{-4}	8.9×10^{-1}
32	4×10^{-3}	5.6

Although the running time of the system is not ideal when ρ is large, this problem will be solved with the development of homomorphic encryption technology.

VII. CONCLUSION

In this paper, we propose a novel matrix notation of automata intended for use in the supervisor encryption framework. We verify the low entropy of this kind of notation and propose algorithms to increase and restore the entropy, respectively. The most significant feature of the proposed framework is the encryption of a supervisor and the signals in the network. Another feature is also essential, that is, the supervisor does not need to keep any private keys to calculate the control input, which means that the supervisor does not require a decryption process inside.

In the future, we are interested in finding new automaton conversion methods to facilitate integration with other encryption schemes. We are also interested in modeling such a system using Petri nets, time Petri nets [52]–[56], or state-tree structures [57], [58]. Another interesting direction for future work is to study in more detail on the framework of the supervisor encryption. Besides, security and machine learning issues in heterogeneous networked systems are of much interest [59]–[65].

ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through research group number RG-1439-009.

REFERENCES

[1] Y. Liu, Y. Peng, B. Wang, S. Yao, and Z. Liu, "Review on cyber-physical systems," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 1, pp. 27–40, Jan. 2017.
[2] F. Yang, N. Wu, Y. Qiao, M. Zhou, and Z. Li, "Scheduling of single-arm cluster tools for an atomic layer deposition process with residency time constraints," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 3, pp. 502–516, Mar. 2017.
[3] Y. Hou, N. Wu, M. Zhou, and Z. Li, "Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 3, pp. 517–530, Mar. 2017.
[4] S. M. M. Rahman, "Cyber-physical-social system between a humanoid robot and a virtual human through a shared platform for adaptive agent ecology," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 190–203, Jan. 2018.
[5] S. Deng, D. Yue, X. Fu, and A. Zhou, "Security risk assessment of cyber physical power system based on rough set and gene expression programming," *IEEE/CAA J. Automatica Sinica*, vol. 2, no. 4, pp. 431–439, Oct. 2015.
[6] S. Haria, "The growth of the hide and seek botnet," *Netw. Secur.*, vol. 2019, no. 3, pp. 14–17, Mar. 2019.
[7] "On the DTrack: A cyber-attack on an Indian nuclear plant raises worrying questions," The Economist, Nov. 2019.
[8] L. Yang, Z. Li, and A. Giua, "Containment of rumor spread in complex social networks," *Inf. Sci.*, vol. 506, pp. 113–130, Jan. 2020.
[9] L. Cao, X. Jiang, Y. Zhao, S. Wang, D. You, and X. Xu, "A survey of network attacks on cyber-physical systems," *IEEE Access*, vol. 8, pp. 44219–44227, 2020.
[10] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2nd ed. Cambridge, MA, USA: MIT Press, 2017.
[11] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York, NY, USA: Springer, 2009.
[12] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989, doi: 10.1109/5.21072.
[13] M. Uma and G. Padmavathi, "A survey on various cyber attacks and their classification," *Int. J. Netw. Secur.*, vol. 15, no. 5, pp. 390–396, 2013.
[14] R. Jacob, J. J. Lesage, and J. M. Faure, "Overview of discrete event systems opacity: Models, validation, and quantification," *Annu. Rev. Control*, vol. 41, pp. 135–146, 2016.
[15] Y. Guo, X. Jiang, C. Guo, S. Wang, and O. Karoui, "Overview of opacity in discrete event systems," *IEEE Access*, vol. 8, pp. 48731–48741, 2020.
[16] L. Garcia, F. Brasser, M. H. Cintuglu, A. R. Sadeghi, O. Mohammed, and S. A. Zonouz, "Hey, my malware knows physics! Attacking PLCs with physical model aware rootkit," in *Proc. NDSS*, San Diego, CA, USA, 2017, pp. 26–28.
[17] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *Proc. CDC*, Osaka, Japan, 2015, pp. 6836–6843.
[18] J. Kim, C. Lee, H. Shim, J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," in *Proc. NECSYS*, Tokyo, Japan, 2016, pp. 175–180.

[19] F. Farokhi, I. Shames, and N. Batterham, "Secure and private control using semi-homomorphic encryption," *Control Eng. Pract.*, vol. 67, pp. 13–20, Oct. 2017.

[20] M. S. Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo, "Towards encrypted MPC for linear constrained systems," *IEEE Control Syst. Lett.*, vol. 2, no. 2, pp. 195–200, Apr. 2018.

[21] R. Fritz, M. Fauser, and P. Zhang, "Controller encryption for discrete event systems," in *Proc. ACC*, Philadelphia, PA, USA, 2019, pp. 5633–5638.

[22] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–180, 1978.

[23] H. Zhou and G. Wornell, "Efficient homomorphic encryption on integer vectors and its applications," in *Proc. ITA*, San Diego, CA, USA, 2014, pp. 1–9.

[24] J. Dyer, M. Dyer, and J. Xu, "Practical homomorphic encryption over the integers for secure computation in the cloud," *Int. J. Inf. Secur.*, vol. 18, no. 5, pp. 549–579, Oct. 2019.

[25] G. Frey, "Design and formal analysis of Petri net based logic control algorithms," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Kaiserslautern, Kaiserslautern, Rheinland-Pfalz, Germany, 2002.

[26] X. Zan, Z. P. Wu, C. Guo, and Z. H. Yu, "A Pareto-based genetic algorithm for multi-objective scheduling of automated manufacturing systems," *Adv. Mech. Eng.*, vol. 12, no. 1, pp. 1–15, 2020, doi: 10.1177/1687814019885294.

[27] D. Sun, Y. Chen, M. A. El-Meligy, M. A. F. Sharaf, N. Wu, and Z. Li, "On algebraic identification of critical states for deadlock control in automated manufacturing systems modeled with Petri nets," *IEEE Access*, vol. 7, pp. 121332–121349, 2019.

[28] D. A. Zaitsev, "Verification of computing grids with special edge conditions by infinite Petri nets," *Autom. Control Comput. Sci.*, vol. 47, no. 7, pp. 403–412, Dec. 2013, doi: 10.3103/S0146411613070262.

[29] Z. Ma, Z. Li, and A. Giua, "Characterization of admissible marking sets in Petri nets with conflicts and synchronizations," *IEEE Trans. Autom. Control*, vol. 62, no. 3, pp. 1329–1341, Mar. 2017.

[30] Z. Ma, Y. Tong, Z. Li, and A. Giua, "Basis marking representation of Petri net reachability spaces and its application to the reachability problem," *IEEE Trans. Autom. Control*, vol. 62, no. 3, pp. 1078–1093, Mar. 2017.

[31] D. You, S. Wang, and C. Seatzu, "Supervisory control of a class of Petri nets with unobservable and uncontrollable transitions," *Inf. Sci.*, vol. 501, pp. 635–654, Oct. 2019.

[32] N. Wu, M. Zhou, and Z. Li, "Short-term scheduling of crude-oil operations: Enhancement of crude-oil operations scheduling using a Petri net-based control-theoretic approach," *IEEE Robot. Autom. Mag.*, vol. 22, no. 2, pp. 64–76, Jun. 2015.

[33] Y. Qiao, N. Wu, F. Yang, M. Zhou, and Q. Zhu, "Wafer sojourn time fluctuation analysis of time-constrained dual-arm cluster tools with wafer revisiting and activity time variation," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 4, pp. 622–636, Apr. 2018.

[34] Q. Zhu, Y. Qiao, and N. Wu, "Optimal integrated schedule of entire process of dual-blade multi-cluster tools from start-up to close-down," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 2, pp. 553–565, Mar. 2019.

[35] N. Q. Wu, M. C. Zhou, L. P. Bai, and Z. W. Li, "Short-term scheduling of crude oil operations in refinery with high-fusion-point oil and two transportation pipelines," *Enterprise Inf. Syst.*, vol. 10, no. 6, pp. 581–610, Sep. 2016.

[36] H. Chen, N. Wu, Z. Li, and T. Qu, "On a maximally permissive deadlock prevention policy for automated manufacturing systems by using resource-oriented Petri nets," *ISA Trans.*, vol. 89, pp. 67–76, Jun. 2019.

[37] Y. H. Hu, Z. Y. Ma, and Z. W. Li, "Design of supervisors for active diagnosis in discrete event systems," *IEEE Trans. Autom. Control*, early access, Jan. 28, 2020, doi: 10.1109/TAC.2020.2970011.

[38] S. Kumar, R. Devaraj, and A. Sarkar, "A hybrid offline-online approach to adaptive downlink resource allocation over LTE," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 3, pp. 766–777, May 2019.

[39] P. M. Lima, M. V. S. Alves, L. K. Carvalho, and M. V. Moreira, "Security against communication network attacks of cyber-physical systems," *J. Control, Autom. Electr. Syst.*, vol. 30, no. 1, pp. 125–135, Feb. 2019.

[40] Y. Y. Liu, K. Cai, and Z. W. Li, "On scalable supervisory control of multi-agent discrete-event systems," *Automatica*, vol. 108, Oct. 2019, Art. no. 108460, doi: 10.1016/j.automatica.2019.06.012.

[41] Q. Chen, L. Yin, N. Wu, M. A. El-Meligy, M. A. F. Sharaf, and Z. Li, "Diagnosability of vector discrete-event systems using predicates," *IEEE Access*, vol. 7, pp. 147143–147155, 2019.

[42] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. STOC*, Bethesda, MD, USA, 2009, pp. 169–178.

[43] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Palo Alto, CA, USA, 2009.

[44] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2014.

[45] C. C. Lee, M. S. Hwang, and S. F. Tzeng, "A new convertible authenticated encryption scheme based on the ElGamal cryptosystem," *Int. J. Found. Comput. Sci.*, vol. 20, no. 2, pp. 351–359, 2009.

[46] M. S. Hwang and C. C. Lee, "Research issues and challenges for multiple digital signatures," *Int. J. Netw. Secur.*, vol. 1, no. 1, pp. 1–6, 2005.

[47] M. S. Hwang, C. C. Lee, and Y. L. Tang, "Two simple batch verifying multiple digital signatures," in *Proc. ICICS*, Xi'an, China, 2001, pp. 233–237.

[48] N. Lv, T. Y. Chen, S. Y. Zhu, J. Yang, Y. Ma, J. W. Jing, and J. Q. Lin, "High-efficiency min-entropy estimation based on neural network for random number generators," *Secur. Commun. Netw.*, vol. 2020, Feb. 2020, Art. no. 4241713.

[49] S. J. Zhao and Q. Y. Zhang, "A unified security analysis of two-phase key exchange protocols in TPM 2.0," in *Proc. TRUST*, Heraklion, Greece, 2015, pp. 40–57.

[50] C. N. Hadjicostis, *Estimation and Inference in Discrete Event Systems*. New York, NY, USA: Springer, 2020.

[51] Y. G. Ramaiah and G. V. Kumari, "Efficient public key homomorphic encryption over integer plaintexts," in *Proc. ISIC*, Yunlin, Taiwan, 2012, pp. 123–128.

[52] L. Li, F. Basile, and Z. Li, "An approach to improve permissiveness of supervisors for GMECs in time Petri net systems," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 237–251, Jan. 2020.

[53] Z. Ding, C. Jiang, and M. Zhou, "Design, analysis and verification of real-time systems based on time Petri net refinement," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 1, pp. 1–18, Jan. 2013.

[54] L. Pan, Z. J. Ding, and M. C. Zhou, "A configurable state class method for temporal analysis of time Petri nets," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 44, no. 4, pp. 482–493, Apr. 2014.

[55] L. Pan, B. Yang, J. Jiang, and M. Zhou, "A time Petri net with relaxed mixed semantics for schedulability analysis of flexible manufacturing systems," *IEEE Access*, vol. 8, pp. 46480–46492, Mar. 2020.

[56] G. Liu, C. Jiang, and M. Zhou, "Time-soundness of time Petri nets modelling time-critical systems," *ACM Trans. Cyber-Physical Syst.*, vol. 2, no. 2, pp. 1–27, Jun. 2018.

[57] D. Wang, X. Wang, and Z. Li, "Nonblocking supervisory control of state-tree structures with conditional-preemption matrices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 3744–3756, Jun. 2020.

[58] C. Gu, X. Wang, and Z. Li, "Synthesis of supervisory control with partial observation on normal state-tree structures," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 984–997, Apr. 2019.

[59] X. Li, E. S. A. Nasr, A. M. El-Tamimi, and Z. Li, "Adaptive consensus of two coupled heterogeneous networked systems with bidirectional actions," *IEEE Access*, vol. 8, pp. 35832–35841, 2020.

[60] P. Zhang, M. Zhou, and G. Fortino, "Security and trust issues in fog computing: A survey," *Future Gener. Comput. Syst.*, vol. 88, pp. 16–27, Nov. 2018.

[61] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 601–614, Feb. 2019.

[62] M. S. Mahmoud and M. M. Hamdan, "Fundamental issues in networked control systems," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 5, pp. 902–922, Sep. 2018.

[63] T. Bai, S. Li, and Y. Zheng, "Distributed model predictive control for networked plant-wide systems with neighborhood cooperation," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 1, pp. 108–117, Jan. 2019.

[64] L. Yang, Z. Yu, M. A. El-Meligy, A. M. El-Sherbeeny, and N. Wu, "On multiplexity-aware influence spread in social networks," *IEEE Access*, vol. 8, pp. 106705–106713, Jun. 2020.

[65] X. Li, Z. Yu, Z. Li, and N. Wu, "Group consensus via pinning control for a class of heterogeneous multi-agent systems with input constraints," *Inf. Sci.*, vol. 542, pp. 247–262, Jan. 2021, doi: 10.1016/j.ins.2020.05.085.

SIAN ZHOU received the B.E. degree in electronic science and technology from the East China University of Technology, Fuzhou, China, in 2012, and the M.E. degree in computer technology from Northwest Normal University, Lanzhou, China, in 2015. He is currently pursuing the Ph.D. degree in computer technology and its applications with the Macau University of Science and Technology, Macao.

His current research interests include security properties, and enforcement of opacity properties in discrete event systems.

ZHENHUA YU received the B.S. and M.S. degrees from Xidian University, Xi'an, China, in 1999 and 2003, respectively, and the Ph.D. degree from Xi'an Jiao Tong University, Xi'an, in 2006. He is currently a Professor with the College of Computer Science and Technology, Institute of Systems Security and Control, Xi'an University of Science and Technology, Xi'an. He has authored more than 20 technical articles for conferences and journals, and holds two invention patents. His research interests include cyber-physical systems, system security, and social networks.

EMAD S. ABOUEL NASR received the Ph.D. degree in industrial engineering from the University of Houston, Houston, TX, USA, in 2005. He is currently a Professor with the Department of Industrial Engineering, College of Engineering, King Saud University, Saudi Arabia, and an Associate Professor with the Department of Mechanical Engineering, Faculty of Engineering, Helwan University, Egypt. His current research interests include CAD, CAM, rapid prototyping, advanced manufacturing systems, and collaborative engineering.

HAITHAM A. MAHMOUD (Senior Member, IEEE) received the Ph.D. degree in industrial engineering from Helwan University, Egypt, in 2012. He worked as an Engineering Consultant for several industrial organizations in Egypt. He is currently an Assistant Professor with the Department of Industrial Engineering, College of Engineering, King Saud University, Riyadh, Saudi Arabia, and the Department of Mechanical Engineering, Faculty of Engineering, Helwan University. His current research interests include optimization modeling, theory, algorithm design with applications in waste management and energy management, financial engineering, and big data.

EMAD MAHROUS AWWAD was born in Menia, Egypt, in 1988. He graduated in 2011. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, King Saud University. He employed as a Teaching Assistant at the Department of Industrial Electronics and Control Engineering, Faculty of Electronic Engineering, Menofia University, Egypt, in 2012. He developed his researches in the field of system design, control of linear and nonlinear systems, embedded system design, the Internet of Things (IOT), and implementation of autonomous mobile robot. His research interests include modeling, optimization, observer design, model predictive controller of vehicle dynamics under the wheel-terrain interaction slippage phenomenon, artificial intelligent, machine learning and deep learning related to the field of robotics, image processing, and renewable energy.

NAIQI WU (Fellow, IEEE) received the B.S. degree in electrical engineering from the Anhui University of Technology, Huainan, China, in 1982, and the M.S. and Ph.D. degrees in systems engineering from Xi'an Jiao Tong University, Xi'an, China, in 1985 and 1988, respectively. From 1988 to 1995, he was with the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. From 1995 to 1998, he was with Shantou University, Shantou, China. He moved to the Guangdong University of Technology, Guangzhou, China, in 1998. He joined the Macau University of Science and Technology, Taipa, Macao, in 2013, where he is currently a Professor with the Institute of Systems Engineering. He is the author or coauthor of one book, five book chapters, and over 160 peer-reviewed journal articles. His research interests include production planning and scheduling, manufacturing system modeling and control, discrete event systems, Petri net theory and applications, intelligent transportation systems, and energy systems. He was an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS. He was an Editor-in-Chief of the *Journal of Industrial Engineering*. He is an Associate Editor of the *Information Sciences*.