mathematics

MDPI

Article

Detection of Actuator Enablement Attacks by Petri Nets in Supervisory Control Systems

Zhenhua Yu [1], Xudong Duan [1], Xuya Cong [1,*], Xiangning Li [2,*] and Li Zheng [3]

[1] College of Computer Science and Technology, Xi'an University of Science and Technology, Xi'an 710054, China
[2] School of Mechano-Electronic Engineering, Xidian University, Xi'an 710071, China
[3] School of Foreign Languages, Changji College, Changji 831100, China
* Correspondence: congxuya@xust.edu.cn (X.C.); lixn@xidian.edu.cn (X.L.)

Abstract: The feedback control system with network-connected components is vulnerable to cyberattacks. We study a problem of attack detection in supervisory control of discrete-event systems. The scenario of a system subjected to actuator enablement attacks is considered in this article. We also consider that some unsafe places that should be protected from an attacker exist in the system, and some controllable events that are disabled by a supervisor might be re-enabled by an attacker. This article proposes a defense strategy to detect actuator enablement attacks and disable all controllable events after detecting an attack. We design algorithmic procedures to determine whether the system can be protected against damage caused by actuator enablement attacks, where the damage is predefined as a set of "unsafe" places. In this way, the system property is called "AE-safe controllability". The safe controllability can be verified by using a basis diagnoser or a basis verifier. Finally, we explain the approach with a cargo system example.

Keywords: Petri nets; discrete event systems; actuator enablement attacks; supervisory control

MSC: 93-02; 93-08; 93C65

Citation: Yu, Z.; Duan, X.; Cong, X.; Li, X.; Zheng, L. Detection of Actuator Enablement Attacks by Petri Nets in Supervisory Control Systems. *Mathematics* **2023**, *11*, 943. https://doi.org/10.3390/math11040943

Academic Editor: Jiangping Hu

Received: 2 January 2023
Revised: 25 January 2023
Accepted: 28 January 2023
Published: 13 February 2023

1. Introduction

The cyber–physical system (CPS) is an intelligent system that integrates communication, control and computing. Safe and supervisory control against potential attacks in cyber–physical systems has drawn extensive attention in recent years [1–7]. To better describe system behaviors, cyber–physical systems are often abstracted as discrete-event systems (DESs). Due to the significance of security concerns in cyber–physical systems, it is necessary to consider attack detection under the framework with supervisory control in discrete-event systems [8,9].

In this article, we explore the issue based on the closed-loop control system shown in Figure 1, where the supervisor controls the system through actuators and sensors. However, the actuators and sensors are often vulnerable to attacks in the process of delivering signals, and attackers can potentially alter the transmitted signals. The object of our study is a discrete-event system driven by events, where the supervisor disables some actuator events according to a given specification. We study the intrusion detection of actuator enablement attacks (AE-attacks) under a closed-loop control system. Specifically, some actuators in the system are vulnerable to intrusion, and an attacker indirectly causes the system to enter an unsafe state by changing control actions of the vulnerable actuators from "disabled" to "enabled".

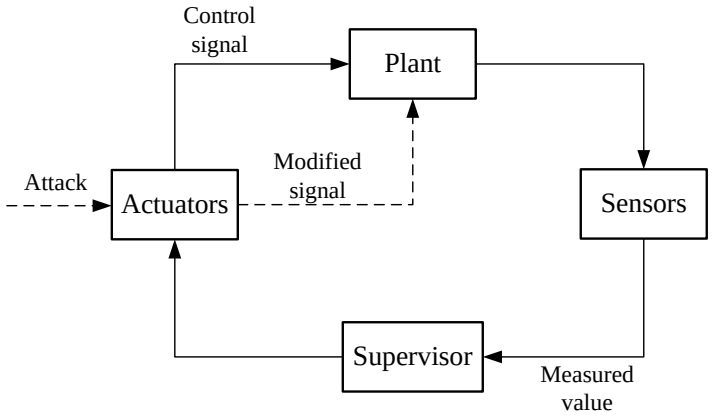

Figure 1. The closed-loop control system architecture.

The study of attack detection in the context of DESs can be traced back as far as the work in [10]. The work in [11] considers a problem of synthesizing a supervisor under removal attacks and sensor insertion attacks. The approach in [12] considers the detection and mitigation of actuator and sensor attacks. In [13], the authors discuss the robust control problem under a sensor replacement attack. The work in [14] investigates integrated sensor deception attacks in the context of DESs. The work in [15,16] focuses on intrusion detection in which the supervisor determines the presence of an intruder by diagnosing faulty behavior in the system. The study in [17,18] presents the issue of supervisory control of DESs under malicious attacks using labeled Petri nets (LPN). In [19], the method of constructing a resilient automaton is proposed by introducing the safety level of the system, which transforms the resilient supervisory synthesis problem into a supervisory control problem. The study in [20] proposes a new attacks mitigation strategy that maximizes the scope of the normal specification while ensuring security. In [21], Rashidinejad et al. outline the existing methods to prevent damage from cyberattacks in cyber–physical systems. The work in [22] investigates joint sensor-actuator network attacks in DESs, defines upper and lower bounds on the language to describe nondeterministic behavior, and successfully solves the issue of supervisory control under network attacks.

The work in [23] proposes a generic attack detection framework with respect to four different types of cyberattacks in supervisory control systems. An automaton model is used to characterize behaviors of systems under attack. Essentially, the use of an automaton model for the description of systems has worked well. However, with the scale of DESs becoming larger and more complex, the state space of the system grows exponentially with the increase of scale, i.e., there is an issue of "state space explosion". The large scale of the system leads to the increase of the probability of failure, and the "state space explosion" also increases the difficulty of fault diagnosis [24]. The aim of our work is to compensate for this drawback and improve the existing detection methods.

The fundamental framework of supervisory control systems in [23] is adopted. However, in contrast to the model in [23], we describe the behavior of a system using a Petri net and construct a basis attack model. More specifically, we replace an automaton with a Petri net and establish a supervisory control system for attack detection of AE-attacks. A basis reachability graph (BRG) is proposed in [25] in which the transitions are divided into two parts, and the net behavior is described by a subset of reachable markings. Our approach is motivated by the research in [25]. However, the work in [25] only classifies events as observable and unobservable. In this article, we use Petri nets to address the issue of attack detection with the AE-attacks, which has never been addressed in the literature. The complexity class of the attack detection problem using Petri nets belongs to the NP-complete problem. Finally, we indicate that only the AE-attacks are considered in this article to present our main results.

Based on the above motivation, this article investigates the detection of AE-attacks by using Petri nets in a control system. In a supervisory control system, there are places

that are unsafe or critical and that should be prevented from being accessed externally, and our goal is to build an attack model by using a Petri net. If the supervisor can block access to unsafe places after an attack, then the system satisfies safe controllability. Note that the traditional framework for detecting AE-attacks for the system using automata has the same purpose as what we do. However, the BRG alleviates the problem of state explosion and is an improvement to the original approach.

The main contributions of the article are outlined as follows:

(1) We modify the existing approach that originally uses automata to describe the system behavior. In the article, we use Petri nets instead of automata. Compared with automata, Petri nets can describe the system behavior in a more compact structure without exhausting the entire state space. Moreover, we use semi-structural approaches to reduce the computational burden in the attack detection problem.

(2) A new approach of constructing the BRG is proposed, where the explanation vector is computed from controllable events, and uncontrollable events are omitted from the state space, making it more efficient to analyze system behavior after an attack has occurred.

This article is divided into five sections. The necessary fundamental knowledge is recalled in Section 2. The notion of AE-attacks and the approach to construct a basis attack model are outlined in Section 3. Section 4 presents the notion of AE-safe controllability and gives algorithms for analyzing AE-safe controllability. Section 5 is mainly concerned with an example of cargo delivery to explain the approach. Section 6 concludes the whole article.

2. Preliminaries

2.1. Basics of Petri Nets

A Petri net (or Petri net structure) is a four tuple $N = (P, T, F, W)$, where P is a finite set of places, T is a finite set of transitions, $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$. We denote by $F \subseteq (P \times T) \cup (T \times P)$ the set of arcs from places to transitions and from transitions to places in the graph. W: $(P \times T) \cup (T \times P) \to \mathbb{N}$ is a mapping that attributes a weight to each arc, where $\mathbb{N}$ is a set of non-negative integers. We denote as ${}^\bullet t = \{p \in P \mid (p, t) \in F\}$ and $t^\bullet = \{p \in P \mid (t, p) \in F\}$ the sets of input places and output places of a transition t, respectively. Similarly, we define ${}^\bullet p = \{t \in T \mid (t, p) \in F\}$ and $p^\bullet = \{t \in T \mid (p, t) \in F\}$. The marking M of a Petri net $N = (P, T, F, W)$ is a mapping from P to $\mathbb{N}$.

A transition t is said to be enabled at a marking M if $\forall p \in {}^\bullet t, M(p) \geq W(p, t)$, denoted as $M[t\rangle$. When firing an enabled transition t, $W(p, t)$ tokens are removed from every input place p of t, and $W(t, p)$ tokens are added to every output place p of t, then generating a new marking $M^{'}$ such that $\forall p \in P, M^{'}(p) = M(p) - W(p, t) + W(t, p)$. Firing t at marking M reaches marking $M^{'}$, denoted as $M[t\rangle M'$. The incidence matrix C of N is a $|P| \times |T|$ integer matrix with $C(p, t) = W(t, p) - W(p, t)$. According to the firing rule of a transition, a transition t is enabled at M, and firing t can reach a marking $M' = M + C(\cdot, t)$. Consequently, for any finite transition sequence σ of a Petri net (N, M_0), we write $M_0[\sigma\rangle M'$ to represent that the sequence of transitions σ is enabled at M_0 and after firing of σ yields M'. The vector $\overrightarrow{\sigma}$ is the Parikh vector of $\sigma \in T^*$ [26], then

$$M' = M_0 + C\overrightarrow{\sigma}. \tag{1}$$

A Petri net is denoted as $G = (N, M_0)$, where M_0 is the initial marking. We use $\mathcal{R}(G)$ to represent the set of all markings that are reached by N from M_0. A net N is bounded if there is an integer $K \in \mathbb{N}$ such that $\forall M \in \mathcal{R}(G)$ and $\forall p \in P, M(p) \leq K$ holds.

2.2. Basis Markings and Basis Reachability Graph

We review several results on basis markings presented in [25,27]. In a basis partition (T_c, T_{uc}), a set T is partitioned into the controllable transition set T_c, and the uncontrollable transition set T_{uc}. C_{uc} is the incidence matrix restricted to $P \times T_{uc}$ and a T_{uc}-induced subnet is a net (P, T_{uc}, F', W'), where F' and W' are the restrictions of F and W, respectively. We denote $|T_c| = n_c$ and $|T_{uc}| = n_{uc}$.

Definition 1. *Given a marking M and a controllable transition $t \in T_c$, we define*

$$\sum(M,t) = \{\sigma \in T_{uc}^* \mid M[\sigma\rangle M', M' \geq Pre(\cdot,t)\} \tag{2}$$

as the set of explanations of t at M, and we define

$$Y(M,t) = \{\boldsymbol{y}_\sigma \in \mathbb{N}^{n_{uc}} \mid \sigma \in \sum(M,t)\} \tag{3}$$

as the set of explanation vectors (or e-vectors).

Therefore, $\sum(M,t)$ is a set of uncontrollable sequences whose firing at marking M can enable transition t. $Y(M,t)$ consists of firing vectors associated with the sequences from $\sum(M,t)$.

Definition 2. *Given a marking M and a transition $t \in T_c$, we define*

$$\sum_{min}(M,t) = \{\sigma \in \sum(M,t) \mid \nexists \sigma' \in \sum(M,t) : \boldsymbol{y}_{\sigma'} \lneq \boldsymbol{y}_\sigma\} \tag{4}$$

as the set of minimal explanations of t at M, and we define

$$Y_{min}(M,t) = \{\boldsymbol{y}_\sigma \in \mathbb{N}^{n_{uc}} \mid \sigma \in \sum_{min}(M,t)\} \tag{5}$$

as the corresponding set of minimal e-vectors.

With the above definitions, a basis marking can be defined as follows. Given a Petri net $G = (N, M_0)$ with its reachability set $\mathcal{R}(G)$, the set of basis markings $\mathcal{M}$ is a subset of $\mathcal{R}(G)$ satisfying: (1) $M_0 \in \mathcal{M}$; (2) $\forall M \in \mathcal{M}$, $\forall t \in T_c$, $\forall \mathbf{y}_{uc} \in Y_{min}(M,t)$, it holds $M' \in \mathcal{M}$, where $M' = M + C_{uc} \cdot \mathbf{y}_{uc} + C(\cdot,t)$.

Briefly, the set of basis markings consists of two parts: an initial marking and the markings reachable from M_0 by firing each controllable transition together with its minimal explanation. All basis markings can be obtained by iterative computation starting from the initial marking M_0.

The BRG generated by a Petri net is a quadruple, denoted by $\mathcal{B} = (\mathcal{M}, \mathcal{T}, \delta, M_0)$, representing a finite state automaton comprised of all basis markings, where: (1) the set $\mathcal{M}$ represents all basis markings; (2) the set $\mathcal{T}$ represents the set of transitions $t \in T_c$; (3) the transition function is denoted as $\delta : \mathcal{M} \times \mathcal{T} \to \mathcal{M}$, i.e., $\delta(M_1,t) = M_2$, $M_2 = M_1 + C_{uc} \cdot \mathbf{y}_{uc} + C(\cdot,t)$, $\mathbf{y}_{uc} \in Y_{min}(M_1,t)$, the function δ can be extended to $\mathcal{M} \times \mathcal{T}^* \to \mathcal{M}$, where $\mathcal{T}^*$ is the Kleene closure of $\mathcal{T}$ [26]; and (4) the state M_0 is the initial marking.

2.3. Supervisory Control Theory

It is assumed that the plant is modeled by a Petri net $G = (N, M_0)$. Assume that $T = T_o \dot{\cup} T_{uo}$, where T_o and T_{uo} represent the sets of observable and unobservable transitions, respectively. Similarly, $T = T_c \dot{\cup} T_{uc}$, where T_c and T_{uc} are the sets of controllable and uncontrollable transitions, respectively. When the behavior of G needs to be restricted for satisfying a specification $\mathcal{K}$, we introduce a feedback control loop as well as a supervisor. The language generated by G is defined by $\mathcal{L}(G) := \{s \in T^* : M_0[s\rangle\}$, which is a set of strings. The natural projection $\mathcal{P}_o : T^* \to T_o^*$ is defined such that: (1) $\mathcal{P}_o(\varepsilon) = \varepsilon$; (2) $\mathcal{P}_o(\omega) = \omega$ if $\omega \in T_o$; (3) $\mathcal{P}_o(\omega) = \varepsilon$ if $\omega \in T_{uo}$; and (4) $\mathcal{P}_o(s\omega) = \mathcal{P}_o(s)\mathcal{P}_o(\omega)$ for $s \in T^*$ and $\omega \in T$, where ε denotes the empty word. Events of the plant are enabled or disabled dynamically by the supervisor, limiting the closed-loop behavior within an acceptable language. Generally, the plant is under partial observation; thus, the supervisor decides to disable certain events on the basis of the projections from strings that are generated by G. To be more specific, a partially observed supervisor is represented as a mapping $S_\mathcal{P} : \mathcal{P}_o[\mathcal{L}(G)] \to 2^T$; the supervisor makes a decision based on $\mathcal{P}_o(s)$ for any string s generated by G. This kind of supervisor is called a $\mathcal{P}$-supervisor. Consequently, when

two different strings s_1 and s_2 have the same projection, they will cause the identical control action.

A sublanguage $\mathcal{K}$ of $\mathcal{L}(G)$ is considered controllable with respect to $\mathcal{L}(G)$ and T_{uc} if $\overline{\mathcal{K}}T_{uc} \cap \mathcal{L}(G) \subseteq \overline{\mathcal{K}}$. Moreover, $\mathcal{K}$ is observable for $\mathcal{L}(G)$, $\mathcal{P}_o$ and T_c if for all $s \in \overline{\mathcal{K}}$ and $\omega \in T_c$, $s\omega \notin \overline{\mathcal{K}}$ and $s\omega \in \mathcal{L}(G)$ implies that $\mathcal{P}_o^{-1}[\mathcal{P}_o(s)]\omega \cap \overline{\mathcal{K}} = \emptyset$. Observability and controllability are essential and sufficient for the presence of a $\mathcal{P}$-supervisor who performs the specification $\mathcal{K}$ [28].

3. Actuator Enablement Attacks

3.1. Attack Definition and Modeling

We graphically depict a control system architecture under attacks in Figure 2. The control system is a plant G controlled by $S_{\mathcal{P}}$. The supervisor monitors the plant events through the projection $\mathcal{P}_o$ generated by the system. Without considering attacks, the closed-loop behavior $\mathcal{L}(S_{\mathcal{P}}/G) = \overline{\mathcal{K}}$, in which $\mathcal{K}$ is an observable with controllable sublanguage of $\mathcal{L}(G)$. $S_{\mathcal{P}}$ is a "nominal" supervisor that is designed to enforce the specification $\mathcal{K}$.

Vulnerable actuators from the supervisor to the plant are frequently attacked. We use $T_{c,v}$ to denote the vulnerable actuator events, which is a subset of all actuator controllable events T_c. Block A_M in Figure 2 represents an attacker model in which the identical observable events can be observed by $\mathcal{P}_o$, and the control actions of the supervisor on vulnerable actuators can be overwritten. In fact, the controllable action affecting plant G is a combination of the controllable behavior of supervisor $S_{\mathcal{P}}$ and attacker A_M on event set T_c. The attack detection module is indicated as D_A. It can also receive the occurrence of observable events by $\mathcal{P}_o$, as the way to infer whether an AE attack has occurred and inform the supervisor $S_{\mathcal{P}}$ when an attack is detected. F_M indicates that the system enters a "defense module" when the supervisor $S_{\mathcal{P}}$ receives the message that the system is attacked. It will disable every controllable event. In this case, the system enters the "defense module", which corresponds to "expect the worst and put safety first". Block U_M denotes that the system enters unmanageable conditions after being attacked.

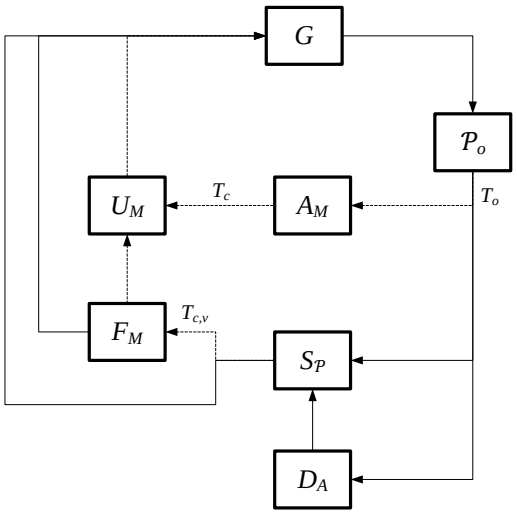

Figure 2. The control system architecture.

The main goal of this article is to build an accurate model for monitoring AE-attacks in the system and to understand the impact of AE-attacks. First, the system is modeled by using a Petri net. Then, basis markings are calculated to obtain a basis attack model, finally a basis diagnoser and a basis verifier are constructed, which are used to judge whether the system satisfies AE-safe controllability. Both methods have their advantages. The flowchart of AE-attack detection is shown in Figure 3.

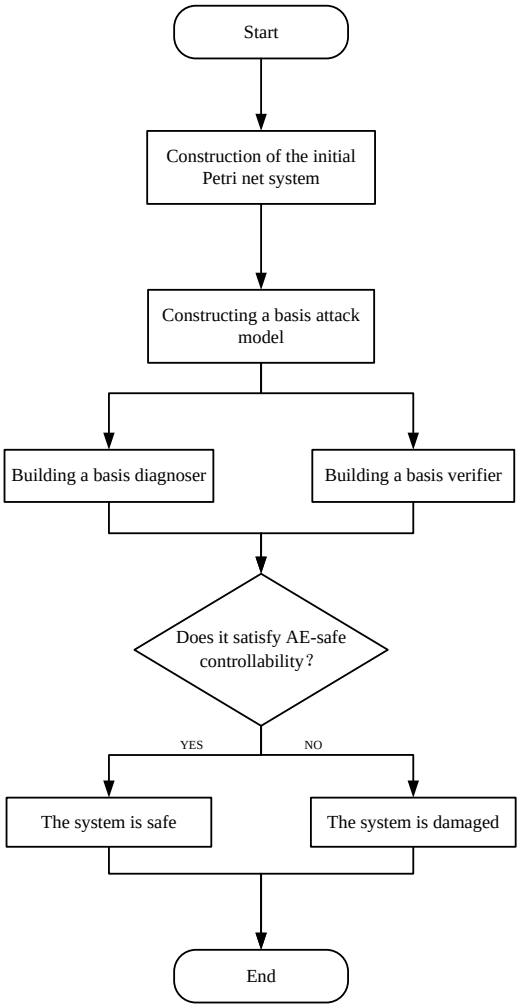

Figure 3. The flowchart of AE-attack detection.

We consider a closed-loop system with vulnerable actuators. The system is modeled as a Petri net $G = (N, M_0)$. To represent the events occurring in a plant, we use the transitions in a Petri net instead, i.e., each event is referred to a transition of a Petri net in the article. When a string s contains an event ω, we write $\omega \in s$. Equivalently, when a string s contains an event in T_c, we write $T_c \in s$. The active event set at place p in G is denoted by $\Gamma_G(p) = \{t \in T : (p,t) \in F\}$.

In particular, the supervisor disables some actuator events to achieve the specification. Then, the attacker intrudes into certain actuators and re-enables these events, overriding the supervisor's control behaviors. The attacker's aim is to make the system arrive at an unsafe state and be damaged through the events that it enables. This type of attack is called an AE attack.

3.2. The Basis Attack Model Under AE-attacks

We consider a Petri net $G = (N, M_0)$, a pair $\pi = (T_c, T_{uc})$ is called a *basis partition* of T, if $(1) T_c \subseteq T$, $T_{uc} = T \setminus T_c$, and (2) the T_{uc} -induced subnet is acyclic. If not, the system will become unstable. In this basis partition, the sets T_c and T_{uc} are called the sets of controllable transitions and uncontrollable transitions, respectively. The controllable events (transitions) may be disabled or enabled by $S_{\mathcal{P}}$. The uncontrollable events are not affected by the supervisor's actions.

We use $T^a_{c,v} = \{\omega^a : \omega \in T_{c,v}\}$ to denote the actuator events intruded by an attacker. We refer to it as the attacked actuator event set and define $T_a = T \cup T^a_{c,v}$. More precisely, ω^a denotes the occurrence of ω that is disabled in the system by the supervisor and then

enabled again by the attacker. The dilation operation is a mapping $D : T^* \to 2^{T_a^*}$ with some properties such as: (1) $D(\varepsilon) = \{\varepsilon\}$, (2) $D(\omega) = \{\omega\}$ if $\omega \in T \setminus T_{c,v}$, (3) $D(\omega) = \{\omega, \omega^a\}$ if $\omega \in T_{c,v}$, and (4) $D(s\omega) = D(s)D(\omega)$ where $s \in T^*$ and $\omega \in T$. We also define the compression operator $\mathbb{C}$: $T_a \to T$ that has the following operating characteristics: (1) $\mathbb{C}(\omega) = \omega$ if $\omega \in T$, and (2) $\mathbb{C}(\omega^a) = \omega$ if $\omega^a \in T_{c,v}^a$. The operator of compression can be extended to $\mathbb{C}$: $T_a^* \to T^*$.

Assumption 1. *The Petri net system used in this paper is a bounded net.*

Assumption 1 means that the method of constructing the basis attack model in this article is applied to a bounded net, since the BRG is finite for a bounded Petri net. Under the condition of bounded nets, the maximum capacity of the places in a Petri net does not exceed a fixed constant K. In this paper, we do not give a specific value of K, which can be an arbitrarily large non-negative integer.

Assumption 2. *All places in the system with uncontrollable transitions do not form a cycle.*

Assumption 2 means that the T_{uc}-induced subnet in the system is acyclic, which allows us to use the state equation to study the reachability of the uncontrollable subnet.

Assumption 3. *The T_{uc}-induced subnet is backward-conflict-free.*

Assumption 3 means that every place has at most one input transition in the T_{uc}-induced subnet. Then, $Y_{min}(M, t)$ is a singleton [25]. Thus, the BRG is considered to be a deterministic finite-state automaton.

We construct a closed-loop system under AE-attacks in Algorithm 1. Let H be the supervisor realized by using a Petri net. Review that a $\mathcal{P}$-supervisor can capture the set of events that are currently enabled. Particularly, enabled unobservable events can be captured with self-loops at the current place in H. More precisely, a supervisor is able to disable some events of G. First, we construct G_a by adding all possible attack behaviors to G with the compression operator $\mathbb{C}$ on $\mathcal{L}(G)$. Specifically, G_a is constructed by adding a parallel transition labeled by $\omega^a \in T_{c,v}^a$ on G. Then, we construct the overall supervisor H_a in the role of AE-attacks. Intuitively, H_a is constructed by adding self-loops to all places with events in $T_{c,v}^a$, when the candidate event's compression is not in the set of active events at the place.

Then, we obtain the closed-loop system under attacks G_M by taking the parallel composition of H_a and G_a. G_M simulates the system behavior in the case where AE-attacks are always presented for all vulnerable actuators. We define a new set of events Φ in the given Petri net, $\Phi = \{t \in T_{uc} \mid \exists p \in P_u, (t, p) \in F\}$, where P_u represents the set of unsafe places. The physical meaning of the set Φ is as follows: the set of basis markings is reachable by firing a sequence σt, where $t \in T_c$ and $\sigma \in T_{uc}^*$. If the sequence σ contains transitions in Φ, then the markings containing unsafe places may not belong to the set of basis markings.

In Algorithm 2, we modify the method of calculating the BRG. Given a marking M and a transition $t \in (T_c \cup \Phi)$, we define $\sum^{\Phi}(M, t) = \{\sigma \in (T \setminus (T_c \cup \Phi))^* \mid M[\sigma\rangle M', M' \geq Pre(\cdot, t)\}$ as the set of explanations of a transition t at a marking M. Correspondingly, the sets of $Y(M, t), \sum_{min}(M, t)$ and $Y_{min}(M, t)$ are modified to the sets of $Y^{\Phi}(M, t), \sum_{min}^{\Phi}(M, t)$ and $Y_{min}^{\Phi}(M, t)$, respectively. The restriction of the incidence matrix to $T_{uc} \setminus \Phi$ is denoted as C_{uc}^{Φ}. Finally, we compute the corresponding basis markings by using the minimal e-vectors and the corresponding transitions in the set $T_c \cup \Phi$ from the initial marking. The basis attack model $G_{\mathcal{B}}$ is constructed subsequently, which allows a more precise analysis of the attacker.

In the resulting basis attack model, states and arcs are drastically reduced as well as the analysis of system behavior becomes more efficient. Since uncontrollable events can always happen at any time, there is no need to display uncontrollable events in the

generated basis attack model $G_{\mathcal{B}}$, and uncontrollable events are used as explanation vectors to fire a certain event.

In the basis attack model $G_{\mathcal{B}}$, only the events in T_c are controllable, and the events in $T_{c,v}$ are the attacker's behaviors. However, the events in $T_{c,v}$ are also controllable, except that the original control action is overwritten by the attacker. The observability of the events in $T_{c,v}$ inherits the observability of the corresponding events in T_c.

Algorithm 1 Algorithm for the closed-loop system under attacks

Input: A Petri net $G = (N, M_0)$ and a supervisor $H = (P_h, T, F_h, M_{0,h}, W_h)$.
Output: A closed-loop system under attacks $G_M = (P_m, T_a, F_m, M_{0,m}, W_m)$.

Let $G_a = (P, T_a, F_a, M_0, W_a)$;
for all $p \in P, \nu \in T_a$ **do**
  if $(p, \mathbb{C}(\nu)) \in F$ **then**
    Let $F_a = F_a \cup \{(p, \mathbb{C}(\nu))\} \cup \{(\mathbb{C}(\nu), \kappa), \forall \kappa \in \mathbb{C}(\nu)^\bullet\}$;
    Let $W_a(p, \mathbb{C}(\nu)) = W(p, \mathbb{C}(\nu))$;
    Let $W_a(\mathbb{C}(\nu), \kappa) = W(\mathbb{C}(\nu), \kappa)$;
  end if
end for
Let $H_a = (P_h, T_a, F_{h,a}, M_{0,h}, W_{h,a})$;
for all $p \in P_h, \nu \in T_a$ **do**
  if $(p, \nu) \in F_h$ **then**
    Let $F_{h,a} = F_{h,a} \cup (p, \nu) \cup \{(\nu, \kappa) \mid (\nu, \kappa) \in F_h\}$;
    Let $W_{h,a}(p, \nu) = W(p, \nu)$;
    Let $W_{h,a}(\nu, \kappa) = W(\nu, \kappa)$;
  else if $(p, \nu) \notin F_h$ **then**
    Let $F_{h,a} = F_{h,a} \cup (p, \nu) \cup (\nu, p)$;
    Let $W_{h,a}(p, \nu) = W(p, \nu)$;
    Let $W_{h,a}(\nu, p) = W(p, \nu)$;
  end if
end for
Compute $G_M = G_a \parallel H_a$;
Output $G_M = (P_m, T_a, F_m, M_{0,m}, W_m)$.

Algorithm 2 Construction of the basis attack model

Input: A closed-loop system under attacks $G_M = (P_m, T_a, F_m, M_{0,m}, W_m)$.
Output: A basis attack model $G_{\mathcal{B}} = (\mathcal{M}, \mathcal{T}, \delta, M_0)$.

Let $\mathcal{M} = \emptyset$, $\mathcal{M}_{new} = \{M_0\}$;
Let T_c be the set of controllable transtions;
while $\mathcal{M}_{new} \neq \emptyset$ **do**
  Select a state $M \in \mathcal{M}_{new}$;
  for all $t \in (T_c \cup \Phi)$ **do**
    Compute $Y^{\Phi}_{min}(M, t)$;
    for all $\mathbf{y} \in Y^{\Phi}_{min}(M, t)$ **do**
      Let $\widehat{M} = M + C^{\Phi}_{uc} \cdot \mathbf{y} + C(\cdot, t)$;
      if $\nexists \widehat{M} \in \mathcal{M} \cup \mathcal{M}_{new}$ **then**
        Let $\mathcal{M}_{new} = \mathcal{M}_{new} \cup \left\{\widehat{M}\right\}$;
      end if
      Let $\delta(M, t) = \hat{M}$;
    end for
  end for
  Let $\mathcal{M} = \mathcal{M} \cup \{M\}$;
  Let $\mathcal{M}_{new} = \mathcal{M}_{new} \setminus \{M\}$;
end while
Output $G_{\mathcal{B}} = (\mathcal{M}, \mathcal{T}, \delta, M_0)$.

Example 1. *The plant G is shown in Figure 4 with $T_c = \{t_1, t_4, t_5, t_7, t_8, t_{10}, t_{11}\}$, $T_{uc} = \{t_0, t_2, t_3, t_6, t_9\}$, $T_{c,v} = \{t_8, t_{10}\}$, $T_o = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{10}, t_{11}\}$, $T_{uo} = \{t_9\}$. The unsafe place in the plant is p_{11}, represented by a square graphic, then $\Phi = \{t_9\}$. The supervisor H is shown in Figure 5, which can control G. The supervisor disables transition t_8 in place p_8, thus stopping the system from arriving at an unsafe place p_{11}. Following Algorithm 1, we build the closed-loop system under attacks by $G_M = G_a \parallel H_a$ in Figure 6. Following Algorithm 2, the basis attack model $G_{\mathcal{B}}$ is computed, starting from the initial marking M_0.*

The generated basis attack model $G_{\mathcal{B}}$ is shown in Figure 7. There are seven states and eight arcs in Figure 7, while there are twelve states and twelve arcs in the reachability graph of the plant. As we can see, since the attacker enables vulnerable events, the system can reach the unsafe place p_{11} through the event t_9.

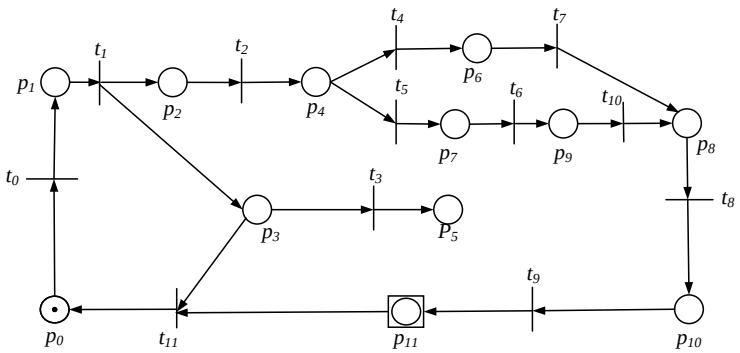

Figure 4. The plant G.

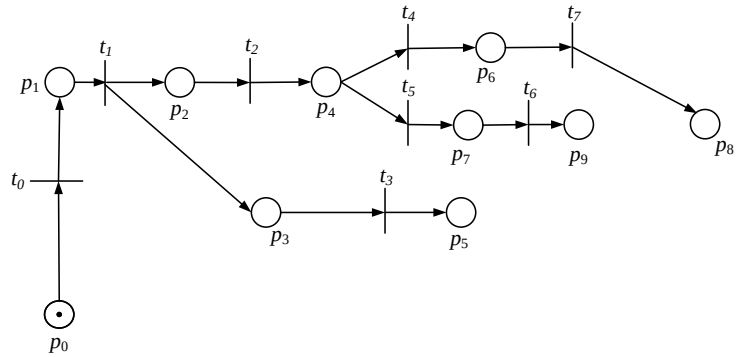

Figure 5. The supervisor H.

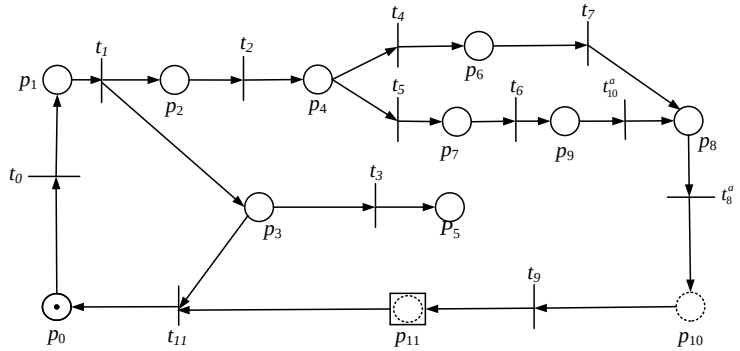

Figure 6. G_M: the closed-loop system under attacks.

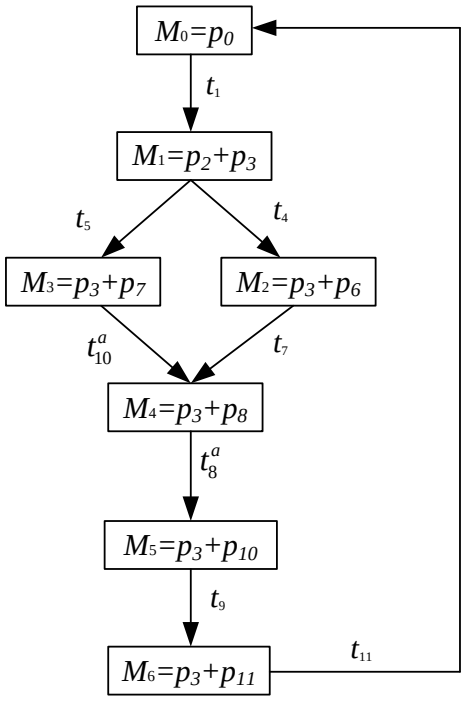

Figure 7. $G_{\mathcal{B}}$: the basis attack model.

4. Detection of Actuator Enablement Attacks

4.1. Detecition Strategy

As mentioned above, under an AE attack, a plant may deviate from the supervisor-enforced specification and arrive at an unsafe place. In order to prevent the impact of such an attack, we design a model for attack detection. When the attack is detected, the system switches to "defense module". This strategy restricts the plant to stop the system from reaching any place in a given set of unsafe places. While it is assumed that every place reached by $S_{\mathcal{P}}/G$ is safe, not all places other than those reached by $S_{\mathcal{P}}/G$ are unsafe. We use P_u to represent the set of unsafe places.

Our techniques are based on those developed in [29] for "safe controllability" and [10] for "disable languages". In particular, using the model built in the above section, we describe the attack detection issue to be a fault diagnosis one in which a fault event is an intrusion event on an actuator that is vulnerable to attacks. An intrusion detection module is designed to monitor the situation of the plant and inform the supervisor when an attack is diagnosed. When a message that the system has been attacked is received from the D_A, the supervisor switches to the "defense module" in which the supervisor disables every controllable event. We point out that attack detection and safe controllability strategies are equally applicable to online implementations, as they rely only on diagnosers and supervisors.

4.2. AE-Safe Controllability

We review the AE-safe controllability in [23]. In particular, the set of unsafe places $P_u \subset P$ is considered. The set of strings with the last event being the vulnerable controllable event is denoted as $\Psi(T_{c,v}) = \{\gamma \in \mathcal{L}(G) : \gamma = \gamma'\omega, \gamma' \in \mathcal{T}^*, \omega \in T^a_{c,v}\}$. The basis attack model $G_{\mathcal{B}}$ generated by Algorithm 2 represents the system behaviors after AE-attacks. Let $\mathcal{M}_u = \{M \in \mathcal{M} \mid \exists p \in P_u : M(p) > 0\}$ be the set of unsafe states in $G_{\mathcal{B}}$. When s' is a rigorous prefix of s, it is written as $s' < s$. Given $L \subseteq T^*$, all subsequent strings starting with s in L are defined as $L/s := \{\gamma : s\gamma \in L\}$. We define $\mathcal{L}(G_{\mathcal{B}}) := \{s \in \mathcal{T}^* : \delta(M_0, s)\text{is defined}\}$ as the language generated by $G_{\mathcal{B}}$. We define $\mathcal{P}^a_o : \mathcal{T}^* \to (T_o \cup D(T_{c,v} \cap T_o))^*$. AE-safe controllability will hold if an attack is detected and successfully stop the plant from arriving at an unsafe state. In the following, we give a definition of AE-safe controllability.

Definition 3. *The basis attack model* $G_{\mathcal{B}} = (\mathcal{M}, \mathcal{T}, \delta, M_0)$ *is from Algorithm 2. Language* $L_{\mathcal{B}} = \mathcal{L}(G_{\mathcal{B}})$ *satisfies AE-safe controllability on projection* $\mathcal{P}_o^a$*, attacked events* $T_{c,v}^a$ *and unsafe states* $\mathcal{M}_u$*, if* $\left(\forall s \in \Psi\left(T_{c,v}^a\right)\right)\left(\forall \gamma \in L_{\mathcal{B}}/s\right)\{(\delta(M_0, s\gamma) \cap \mathcal{M}_u \neq \emptyset) \wedge (\forall s' < s\gamma, \delta(M_0, s') \cap \mathcal{M}_u = \emptyset)\}$ $\Rightarrow (\exists \gamma_1, \gamma_2 \in \mathcal{T}^*)\left[(\gamma = \gamma_1\gamma_2) \wedge \left((\nexists \mu \in L_{\mathcal{B}})\left[\mathcal{P}_o^a(s\gamma_1) = \mathcal{P}_o^a(\mu) \wedge T_{c,v}^a \notin \mu\right]\right) \wedge (T_c \in \gamma_2)\right]$.

Briefly, we claim that the system satisfies AE-safe controllability if $L_{\mathcal{B}}$, $\mathcal{P}_o^a$ and $\mathcal{M}_u$ are understood and Definition 3 holds. The definition of AE-safe controllability is illustrated in Figure 8. In the above definition, the state is first arrived via the string s, whose final event $\omega_{c,v}^a$ of that string s is the actuator event under attacks. The string γ is a continuation of s that first reaches the unsafe state. A system satisfies AE-safe controllability if for every pair of s and γ, where γ can be divided as $\gamma = \gamma_1\gamma_2$ and where: (1) after $s\gamma_1$, an attacked event can be diagnosed, and (2) there exists a controllable event in γ_2. Review above that every event in T_c is controllable, while the attacked events in it are uncontrollable. In other words, AE-safe controllability will hold if an attack is detected, then disable a controllable event and successfully stop the plant from arriving at an unsafe state; this controllability holds true if each attack cannot be missed. We should point out that the detection requirements after string $s\gamma_1$ is that an attack is detected at each vulnerable actuator (cf. $T_{c,v}^a \notin \mu$ in Definition 3). The module D_A will inform $S_{\mathcal{P}}$ to disable all controllable actuator events after it is sure that an attack has been detected. The construction method of $G_{\mathcal{B}}$ and the requirements to determine AE-safe controllability yield the results shown below.

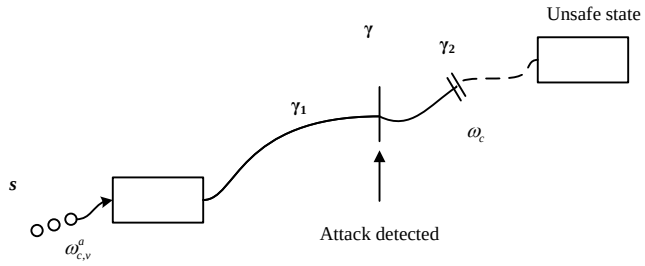

Figure 8. Graphic representation of AE-safe controllability.

Theorem 1. *Considering AE-attacks and the "defense module", the plant G does not arrive at an unsafe state if and only if it satisfies AE-safe controllability.*

Proof of Theorem 1. ($\Rightarrow$) By contradiction, suppose that the plant will still arrive at an unsafe state while satisfying AE-safe controllability. By Definition 3, if the system reaches an unsafe state, it will violate the safety controllability requirement, which contradicts the assumption.

($\Leftarrow$) By contradiction, supposing that the system does not satisfy the safety controllability requirement, but also does not reach the unsafe state, according to Definition 3, if the safety controllability is violated, the system reaches the unsafe state and suffers damage, which contradicts the assumption. □

4.3. Test of AE-Safe Controllability Using Basis Diagnoser

To determine whether a system satisfies AE-safe controllability, we formulate an algorithm for constructing a diagnoser using the basis attack model. The diagnoser depends on the calculation of an automaton observer, which is generated by a parallel composition of a plant automaton and a label automaton, as mentioned in [26,30]. We propose algorithms to verify that the module D_A is able to detect attacks before the system arrives at an unsafe state and the supervisor can disable controllable events to stop the system from arriving at P_u. We first review the definition of first-entered certain states as described in [29]. The diagnoser and related terms are explained in [26].

Definition 4. *The basis diagnoser is described as* $G_D = (\mathcal{M}_d, T_o, F_d, M_{0,d})$*, which consists of a combination of the basis attack model* $G_{\mathcal{B}}$ *and the label automaton* A_ξ*. Three new sets are defined as follows:* $Q_U = \{q \mid q \in \mathcal{M}_d : q \text{ is uncertain}\}$*,* $Q_N = \{q \mid q \in \mathcal{M}_d : q \text{ is normal}\}$*, and* $Q_Y = \{q \mid q \in \mathcal{M}_d : q \text{ is certain}\}$*. The set of first-entered certain states is* $\mathcal{FC} = \{q \mid q \in Q_Y : (\exists q' \in Q_U \cup Q_N, \exists \omega \in T_o)[F_d(q', \omega) = q]\}$.

First, we build the label basis attack model G_ξ with Algorithm 3. It can be seen by the construction of $G_{\mathcal{B}}$ that our aim is to determine the course of events in $T^a_{c,v}$, according to the set of observable events. Specifically, the events in $T^a_{c,v}$ are all "fault" events to be detected, and they are considered to be the identical fault type. Thus, we want to build a basis diagnoser G_D. In Algorithm 3, we construct a label basis attack model $G_\xi = (\mathcal{M}_\xi, \mathcal{T}_\xi, \delta_\xi, M_{0,\xi})$ by using the label automaton A_ξ in Figure 9 and the attacked actuator events in $T^a_{c,v}$.

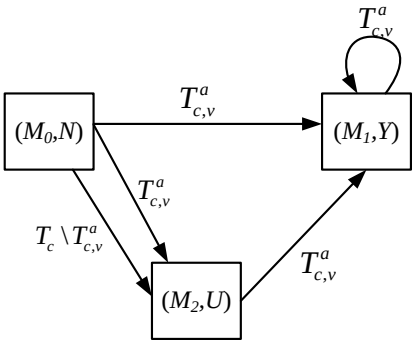

Figure 9. Label automaton A_ξ.

Algorithm 3 Building a label basis attack model

Input: Basis attack model $G_{\mathcal{B}} = (\mathcal{M}, \mathcal{T}, \delta, M_0)$
Output: Label basis attack model $G_\xi = (\mathcal{M}_\xi, \mathcal{T}_\xi, \delta_\xi, M_{0,\xi})$

Sign the initial marking M_0 as "N";
for all $M_c \in \mathcal{M}$ **do**
  for all $t \in \mathcal{T}$ **do**
    if $\delta(M_p, t) = M_c$ **then**
      if M_p is labeled with "U" **then**
        Sign M_c as "U";
      else if M_p is labeled with "Y" or $t \in T_{c,v}$ **then**
        Sign M_c as "Y";
        for all $t' \in \mathcal{T}$ and $t' \neq t$ **do**
          if $\delta(M'_p, t') = M_c$ and $t' \notin T_{c,v}$ and M'_p is not labeled with "Y" **then**
            Sign M_c as "U";
          end if
        end for
      else
        Sign M_c as "N";
      end if
    end if
  end for
end for
Output $G_\xi = (\mathcal{M}_\xi, \mathcal{T}_\xi, \delta_\xi, M_{0,\xi})$.

Next, we introduce Algorithm 4, which is a diagnoser-based algorithm to verify AE-safe controllability. We start constructing a basis diagnoser $G_D = Obs(G_\xi, T_{a,uo})$, where $Obs(G_\xi, T_{a,uo})$ represents the observer of G_ξ about the unobservable event set $T_{a,uo}$ with $T_{a,uo} = T_{uo} \cup D(T_{c,v} \cap T_{uo})$. In Step 2, we examine each uncertain state to determine

whether it contains an unsafe state, and if so, the diagnoser will fail to detect the occurrence of an attack before the system arrives at an unsafe state, thus violating AE-safe controllability. Then, we calculate the set $\mathcal{FC}$ and examine each state in $\mathcal{FC}$ to determine whether it contains an unsafe state in Step 6. If so, even though an attack is detected, the system has already arrived at an unsafe state; therefore the system violates AE-safe controllability. In Step 9, we consider a set of events $T' \subseteq T$ and a state $M \in \mathcal{M}$. The set of reachable states for T' and M is $Reach(G_{\mathcal{B}}, M, T') = \{M' \in \mathcal{M} : (\exists s \in (T')^*)[\delta(M, s) = M']\}$. Finally, the set of reachable states is found from $\mathcal{FC}$ by attacked actuator events or uncontrollable events. Then, the states in the set are examined at Step 10 to determine whether they contain unsafe states. If so, even if an attack is diagnosed at this time, it cannot stop the system from arriving at an unsafe state. Thus, it does not satisfy AE-safe controllability. In Algorithm 4, the projection of q on the corresponding state set of $G_{\mathcal{B}}$ is denoted as $q_{\downarrow M} := \{M : (\exists l)[(M, l) \in q]\}$.

Algorithm 4 AE-safe controllability test using basis diagnoser

Inputs: • G_{ξ}: Label basis attack model
• $\mathcal{M}_u$: set of unsafe states
• $T^a_{c,v}$: set of attacked actuator events
Output: AE-Safe Controllability $\in \{true, false\}$
1: The basis diagnoser $G_D = Obs(G_{\xi}, T_{a,uo})$;
2: **if** there is uncertain state $q = \{(M_{i_1}, \xi_{i_1}) \cdots (M_{i_n}, \xi_{i_n})\}$ in which there exists $M_{i_j} \in \mathcal{M}_u$ **then**
3: AE-safe controllability=false;
4: **else**
5: Compute $\mathcal{FC}$ according to Definition 4;
6: **if** there is $q = \{(M_{i_1}, Y) \cdots (M_{i_n}, Y)\}$ in which there exists $M_{i_j} \in \mathcal{M}_u$ **then**
7: AE-safe controllability=false;
8: **else**
9: Compute $\mathcal{M}^{uc} = \bigcup_{q \in \mathcal{FC}} \bigcup_{M_x \in q_{\downarrow M}} Reach(G_{\mathcal{B}}, M_x, T^a_{c,v} \cup \Phi)$;
10: **if** $\mathcal{M}^{uc} \cap \mathcal{M}_u \neq \emptyset$ **then**
11: AE-safe controllability=false;
12: **else**
13: AE-safe controllability=true.
14: **end if**
15: **end if**
16: **end if**

Proposition 1. *Consider $G_{\mathcal{B}} = (\mathcal{M}, \mathcal{T}, \delta, M_0)$ from Algorithm 2. The basis diagnoser G_D is constructed in Algorithm 4. Language $L_{\mathcal{B}}$ does not satisfy AE-safe controllability for $\mathcal{P}^a_o$, $T^a_{c,v}$, and $\mathcal{M}_u$ if and only if any of these conditions are true:*

(1) There exists $q_U = \{(M_{i_1}, \xi_{i_1}), \cdots, (M_{i_n}, \xi_{i_n})\} \in Q_U$ such that $\exists j \in \{1, \cdots, n\}$, $M_{i_j} \in \mathcal{M}_u$ and $\xi_{i_j} = Y$; (2) There exists $q_Y = \{(M_{i_1}, Y), \cdots, (M_{i_n}, Y)\} \in \mathcal{FC}$ such that $\exists j \in \{1, \cdots, n\}$, $M_{i_j} \in \mathcal{M}_u$; (3) There exists $M_x \in \mathcal{M}^{uc}$ such that $M_x \in \mathcal{M}_u$, where $\mathcal{M}^{uc}$ is defined in Algorithm 4.

Proof of Proposition 1. ($\Rightarrow$) Suppose the plant $L_{\mathcal{B}}$ is AE-safe controllable; there exists $\left(M_{i_j}, \xi_{i_j}\right)$ with $\xi_{i_j} = Y$, $M_{i_j} \in \mathcal{M}_u$ in the set of q_U. It indicates that the system has detected an attack occurrence at this time, while the plant has reached the unsafe state, According to Definition 3, it is known that AE-safe controllability is violated, which is a contradiction. The remaining two conditions are the same as above.

($\Leftarrow$) By contradiction, suppose that there is no $\left(M_{i_j}, \xi_{i_j}\right)$ with $\xi_{i_j} = Y$, $M_{i_j} \in \mathcal{M}_u$ in the set of q_U, and the plant $L_{\mathcal{B}}$ violates AE-safe controllability. According to Definition 3, if the plant does not satisfy AE-safe controllability, $L_{\mathcal{B}}$ has reached an unsafe state $M_{i_j} \in \mathcal{M}_u$

after attacks, which is a contradiction. The remaining two conditions are the same as above. □

Note that since the event ω_a is observable, the basis diagnoser can detect an attack immediately when it occurs on the vulnerable actuator events in $\omega \in T_{c,v} \cap T_o$. In such a case, the system might still arrive at an unsafe state and violate AE-safe controllability via the attacked actuator events.

Example 2. *Based on Example 1, the basis attack model $G_{\mathcal{B}}$ is shown in Figure 7. Next, we verify that the system satisfies AE-safe controllability according to Algorithm 4. In the first step, we construct the label basis attack model G_{ξ} on $T^a_{c,v} = \{t^a_8, t^a_{10}\}$ in Figure 10. For convenience, assuming $T_{a,uo} = \emptyset$, the basis diagnoser G_D is the identical graph as G_{ξ}. By checking states of the diagnoser in Figure 10, we can find that an attacked actuator event will be detected in (M_5, Y) before the system arrives at the unsafe state M_6. Finally, we can see that $\mathcal{M}^{uc} = \{M_5, M_6\}$ contains the unsafe state M_6 at Step 10, which means that although an attack can be detected in advance by the diagnoser, the plant can still enter the unsafe state $M_6 \in \mathcal{M}_u$ under the attack since event t_9 is uncontrollable, thus violating AE-safe controllability.*

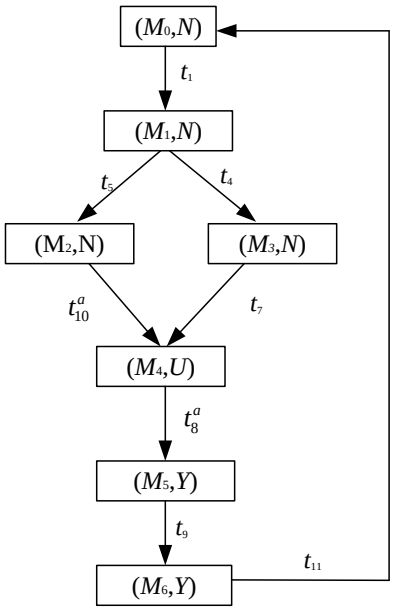

Figure 10. Label basis attack model G_{ξ}.

4.4. Test of AE-Safe Controllability Using Basis Verifier

This subsection verifies AE-safe controllability by using a basis verifier, which is another diagnostic method. The simple verifier-based approach was proposed and used in [31–34]. Compared to the diagnoser presented in the above section, the verifier requires lower complexity, but the verifier is not as suitable for online diagnosis as the diagnoser. Both methods are suitable for different scenarios, and both have their own advantages.

Algorithm 5 shows in detail how AE-safe controllability can be tested by a basis verifier. In the first step, the label basis attack model G_{ξ} is constructed by using Algorithm 3. In the second step, the basis verifier G_V is constructed by using the method of [31]. G_V is obtained by computing G_N and G_F that denote the model of normal and faulty behavior, respectively. At a state M, the active event set of G_V is denoted as $\Gamma_V(M)$. In the process of constructing G_N, the state space is represented by $\mathcal{M}_N$, and then the unobservable events are renamed using the renaming function R: $\mathcal{T} \setminus T^a_{c,v} \to T_R$, where $R(\omega) = \omega$, if $\omega \in T_{a,o}$ and $R(\omega) = \omega_R$, if $\omega \in T_{a,uo} \setminus T^a_{c,v}$. The observable event set is $T_{a,uo} = \{T_{uo} \cup D(T_{c,v} \cap T_{uo})\}$. Therefore, the unobservable events of G_N and G_F are considered to be "private" events. In Step 4, all states in G_V are judged for the presence of unsafe states, and if they exist, AE-safe

controllability is violated. In Step 7, we present a new state set A that indicates the states reached by the remaining observable events, which may contain unsafe states, and then add a self-loop of uncontrollable events under A. After diagnosing the attack, the system violates AE-safe controllability if there is a path to reach the unsafe state only by the unobservable events. At Step 11, we compute the combined basis verifier $G_T = G_V^{cd} \parallel G_F$, whose state space is represented by $\mathcal{M}_T$. In Step 12, if there is an unsafe state in G_T, the unsafe state was reached before the attack was detected.

Proposition 2. *Let $L_{\mathcal{B}}$ be the language generated by $G_{\mathcal{B}}$. Then, $L_{\mathcal{B}}$ does not satisfy AE-safe controllability with respect to $\mathcal{P}_o^a$: $\mathcal{T}^* \to T_{a,o}^*$, $T_{c,v}^a$ and $\mathcal{M}_u$ if and only if any of these conditions is true: (1) There exists $M_V = \{(M_N, N), (M, Y)\} \in \mathcal{M}_V$ such that $M \in \mathcal{M}_u$, where $M_N \in \mathcal{M}_N$ and $M \in \mathcal{M}$; (2) There exists $\left\{M_V^{cd}, (M, Y)\right\} \in \mathcal{M}_T$ such that $M_V^{cd} = A$ and $M \in \mathcal{M}_u$, where $M_V^{cd} \in \mathcal{M}_V^{cd}$ and $M \in \mathcal{M}$.*

Proof of Proposition 2. ($\Rightarrow$) By contradiction, let $L_{\mathcal{B}}$ be AE-safe controllable. Then, there exists $M_V = \{(M_N, N), (M, Y)\} \in \mathcal{M}_V$, $M \in \mathcal{M}_u$. It means that the plant has detected an attack occurrence at this time, while the plant has reached the unsafe state. According to Definition 3, it is known that the AE-safe controllability is violated, which is a contradiction. The remaining condition is the same as above.

($\Leftarrow$) By contradiction, suppose that there is no $M_V = \{(M_N, N), (M, Y)\} \in \mathcal{M}_V$, $M \in \mathcal{M}_u$ and $L_{\mathcal{B}}$ that violates AE-safe controllability. According to Definition 3, if the plant does not satisfy AE-safe controllability, it indicates that the plant has reached an unsafe state $M \in \mathcal{M}_u$ after an attack, which is a contradiction. The remaining condition is the same as above. □

Algorithm 5 AE-safe controllability test using basis verifier

Inputs: • $G_{\mathcal{B}} = (\mathcal{M}, \mathcal{T}, \delta, M_0)$: basis attack model
• $\mathcal{M}_u$: set of unsafe states
• $T_{c,v}^a$: set of attacked actuator events
Output: AE-Safe Controllability$\in \{true, false\}$
1: Build G_ξ according to Algorithm 3;
2: Build basis verifier $G_V = (\mathcal{M}_V, T_R \cup \mathcal{T}, F_V, M_{0,V})$ assuming $T_{c,v}^a$ be the set of fault events according to Algorithm 1 in [31];
3: Let $\Gamma_V(M) = \{t \in T_R \cup \mathcal{T} \mid \exists M' \in \mathcal{M}_V, F_V(M, t) = M'\}$;
4: **if** there exists $\{(M_N, N), (M, Y)\}$ of G_V such that $M \in \mathcal{M}_u$ **then**
5: Safe Controllability=false;
6: **else**
7: Build $G_V^{cd} = \left(\mathcal{M}_V^{cd}, T_R \cup \mathcal{T}, F_V^{cd}, M_{0,V}\right)$, where
8: • $\mathcal{M}_V^{cd} = \mathcal{M}_V \cup \{A\}$;
9: • $F_V^{cd}(M_V, \tau) = F_V(M_V, \tau)$, if $\tau \in \Gamma_V(M_V)$;
10: • $F_V^{cd}(M_V, \tau) = A$, if $\tau \in T_{a,o} \wedge \tau \notin \Gamma_V(M_V)$;
11: • $F_V^{cd}(A, \tau) = A$ for all $\tau \in \Phi \cup T_{c,v}^a$;
12: Build $G_T = G_V^{cd} \parallel G_F$, where G_F is defined in Algorithm 1 in [31];
13: **if** there exists $\left\{M_V^{cd}, (M, \xi)\right\}$ in G_T such that $M_V^{cd} = A$ and $M \in \mathcal{M}_u$ **then**
14: Safe Controllability=false;
15: **else**
16: Safe Controllability=true.
17: **end if**
18: **end if**

Example 3. *Again reviewing the system in Example 1, the basis attack model $G_{\mathcal{B}}$ is shown in Figure 7 in which the controllable events observable events and vulnerable events are $T_c = \{t_1, t_4, t_5, t_7, t_8, t_{10}, t_{11}\}$, $T_o = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_{10}, t_{11}\}$, $T_{c,v} = \{t_8, t_{10}\}$, respectively.*

The model G_N, which represents the normal behavior of the system, is shown in Figure 11a. The model G_F, which represents the fault/attacked behavior, is depicted in Figure 11b, and the basis verifier G_V is displayed in Figure 12. By Step 8 of Algorithm 5, a new state A needs to be added to G_V, which is added into the basis verifier under attacks G_V^{cd}. Each state in G_V is linked to state A by observable events except the self-loop of uncontrollable events under state A. The basis verifier under attacks G_V^{cd} is shown in Figure 13. After that, G_T is obtained by calculating $G_V^{cd} \parallel G_F$, as shown in Figure 14. According to the judgment condition of Step 13 in Algorithm 5, it is known that the system violates AE-safe controllability, since the state $\{A, (M_6, Y)\}$ in G_T consists of two parts, state A and $M_6 \in \mathcal{M}_u$.

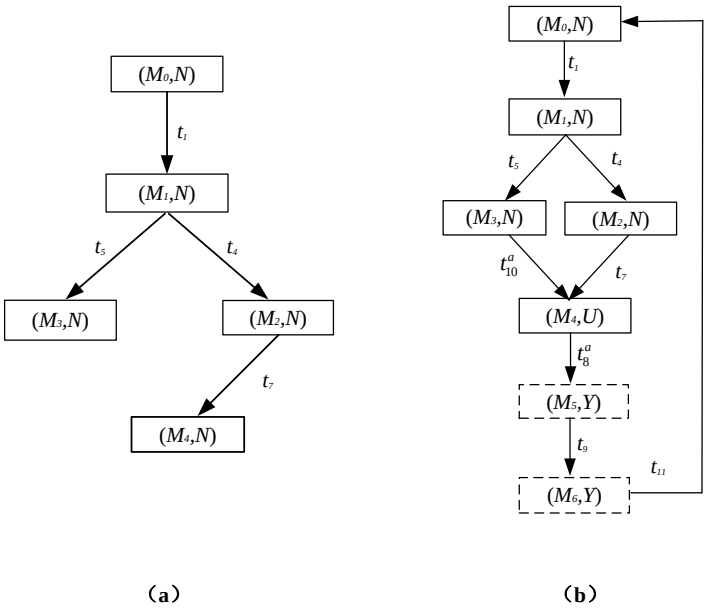

Figure 11. (**a**) The system model under normal behavior G_N and (**b**) the system model under attacked behavior G_F.

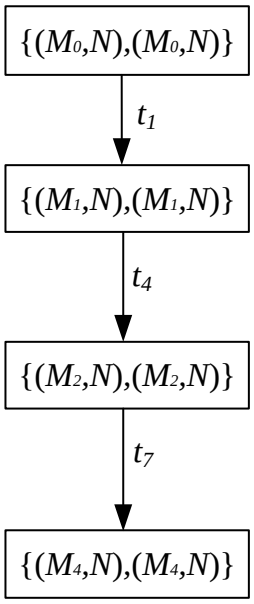

Figure 12. The basis verifier G_V.

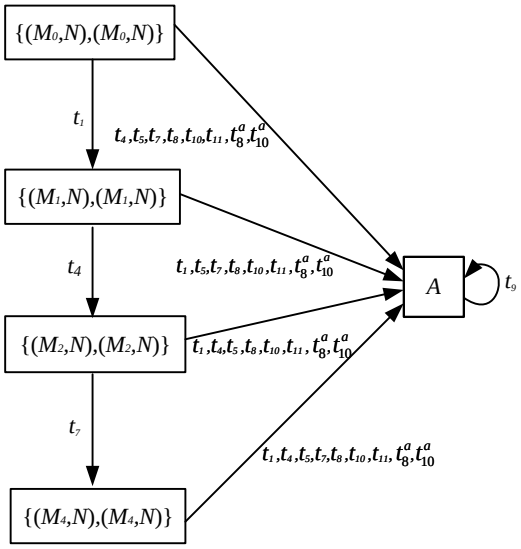

Figure 13. The basis verifier under attacks G_V^{cd}.

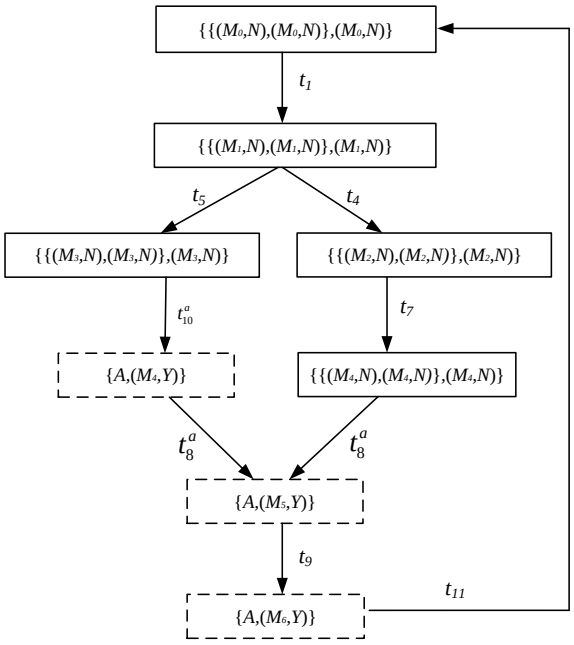

Figure 14. The combined basis verifier G_T.

5. Computational Efficiency Analysis and Experiments

First, we consider the complexity of constructing a system network model by using a Petri net. In this case, the relationship between the construction of the Petri net model and the size of the actual system is linear. For the complexity of constructing a BRG, if all the transitions in the system are controllable, then the basis marking set and the reachable marking set are the same, i.e., $\mathcal{M} = \mathcal{R}(G)$. Therefore, in the worst case, constructing a BRG has the same complexity as constructing a reachable graph. In this case, the complexity is exponential with respect to the number of places and the initial marking. Next, we consider the complexity of building the basis attack model. Since the basis attack model is built based on BRG, the complexity of constructing the basis attack model is identical to the BRG. Finally, in the detection of attacks using the basis diagnoser, the complexity of building the basis diagnoser is exponential with respect to the number of states in the basis attack model, and the basis verifier requires polynomial time with respect to the state space of the BRG.

For the attack detection method under the basis attack model mentioned above, we give some numerical examples to validate the construction method and study the efficiency of the model by experiments comparing the number of states of the basis attack model with the number of states of the reachable graph. Finally, we determine whether AE-security controllability is satisfied. From the experiments, the number of states under a basis attack model is significantly reduced. The experiments are simulated on a laptop computer with Core-i5 2.40 GHz/2.50 GHz CPU using *Petri Net Basis Reachability Space Generator* [35]. The experimental comparison is shown in Table 1.

Table 1. Experimental comparison of the basis attack model and the traditional attack model [23].

Experimental Index	Percentage of Controllable Transitions	Number of Basis Markings	Number of Reachable Markings	Marking Compression Rate	AE-Safe Controllability
1	100%	16	16	0%	True
2	97%	57	84	32%	False
3	81%	58	72	19%	True
4	62%	42	72	41%	True
5	60%	39	81	51%	False
6	51%	33	112	70%	True
7	43%	23	124	81%	False
8	35%	23	112	79%	False

6. Example

A cargo transportation system under an AE attack is considered, which is represented by a Petri net, as depicted in Figure 15. The system is responsible for warehousing, processing, handling, packing and discharging cargoes from the factory. The places indicate the location in the factory, the transitions indicate intelligent automatic processing machines, and the arrow indicates the conveyor belt. The attacker modifies the actuator information to force the machine to start, whose ultimate goal is to steal secret information when the system reaches the unsafe state p_{33}. The first batch of cargo enters the factory at p_0, the quality check is performed at t_2, the unqualified products are sent to p_4, and the qualified products continue to be sent to p_3 for the next check. The processing route is selected at p_6 according to the number of cargoes. If the number is small, then t_9 is enabled, otherwise t_8 is enabled. The system arrives at p_{12} for classification, t_{13} is normally enabled for processing, t_{12} is the alternate processing route, and it finally arrives at p_{19}. When it arrives at p_{22}, t_{23} is enabled to select the method of cargo transportation and print the transportation order. Expedited transportation arrives at p_{23}, and ordinary transportation arrives at p_{24}. When the system arrives at p_{33}, the factory scans each cargo and registers the information in the cloud platform. In the whole system, p_{33} is the most critical step and the most vulnerable to intrusion, and the attacker wants to steal the secret information at p_{33}. The goods are finally released at p_{34}. After a shipment is completed, it reverts to p_0 for the next shipment.

The set of controllable events of the plant is $T_c = \{t_1, t_2, t_5, t_7, t_8, t_9, t_{17}, t_{18}, t_{19}, t_{22}, t_{23}, t_{28}, t_{33}, t_{34}\}$, the set of remaining events $T_{uc} = T \setminus T_c$ is uncontrollable, and the set of vulnerable events is $T_{c,v} = \{t_{17}, t_{28}\}$. The set of observable events is $T_o = \{t_0, t_1, t_2, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}, t_{27}, t_{28}\}$. The set of unsafe places is $P_u = \{p_{33}\}$. The set Φ is $\Phi = \{t_{32}\}$. Let system G be controlled by the supervisor $S_{\mathcal{P}}$, who always disables the set of events $T_{c,v}$ in order to prevent damage to the system.

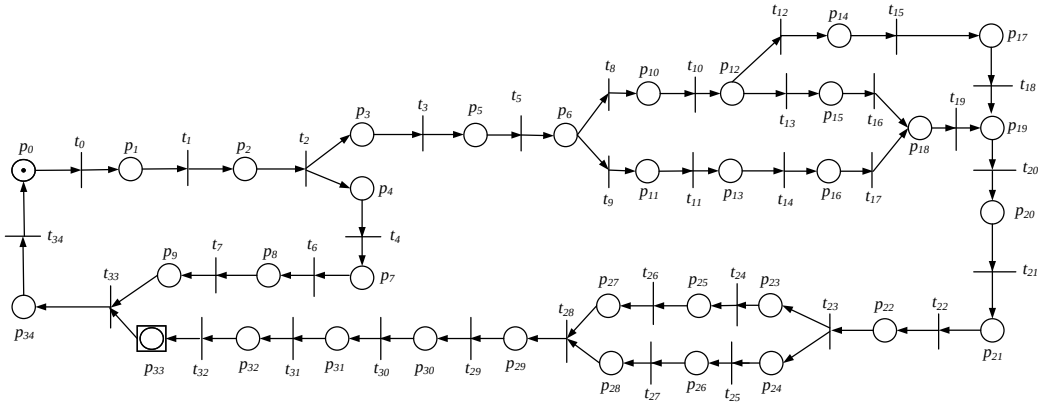

Figure 15. Petri net G.

We consider that $\mathcal{K} \subset \mathcal{L}(G)$ is an observable and controllable behavior which is realized by $S_{\mathcal{P}}$. The realization H of the supervisor is depicted in Figure 16. According to Algorithm 1, we first construct G_a to represent the change of system state after being attacked. Next, we build H_a to represent the supervisor after being attacked. Then, we construct the closed-loop system G_M under attacks by using parallel composition, as shown in Figure 17. Finally, according to the algorithm presented above, the basis attack model is generated, as depicted in Figure 18. We can see that the set of unsafe states is $\mathcal{M}_u = \{M_{20}, M_{21}\}$. After the attacked event t_{28}, the plant state changes from M_{17} to M_{19}, and since the events in the set Φ are uncontrollable events, the system state continues to run uncontrollably from M_{19} to the unsafe state M_{21}.

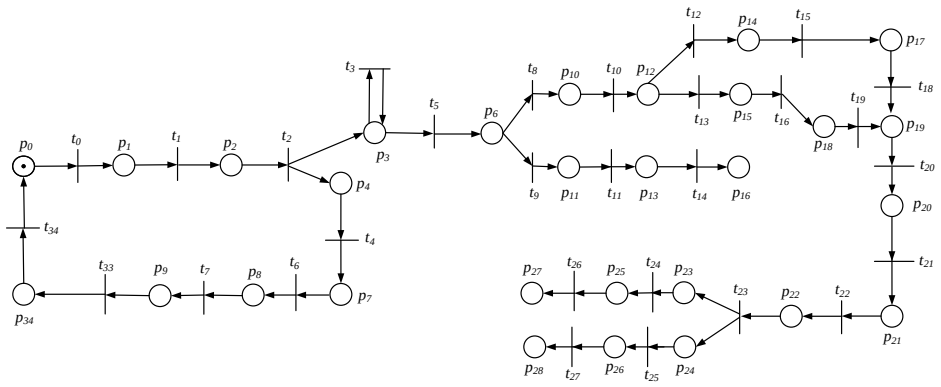

Figure 16. Supervisor H.

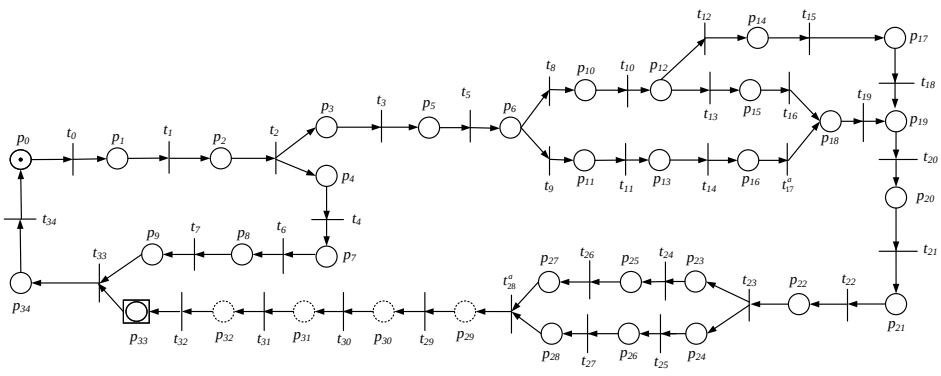

Figure 17. The closed-loop system under attacks G_M.

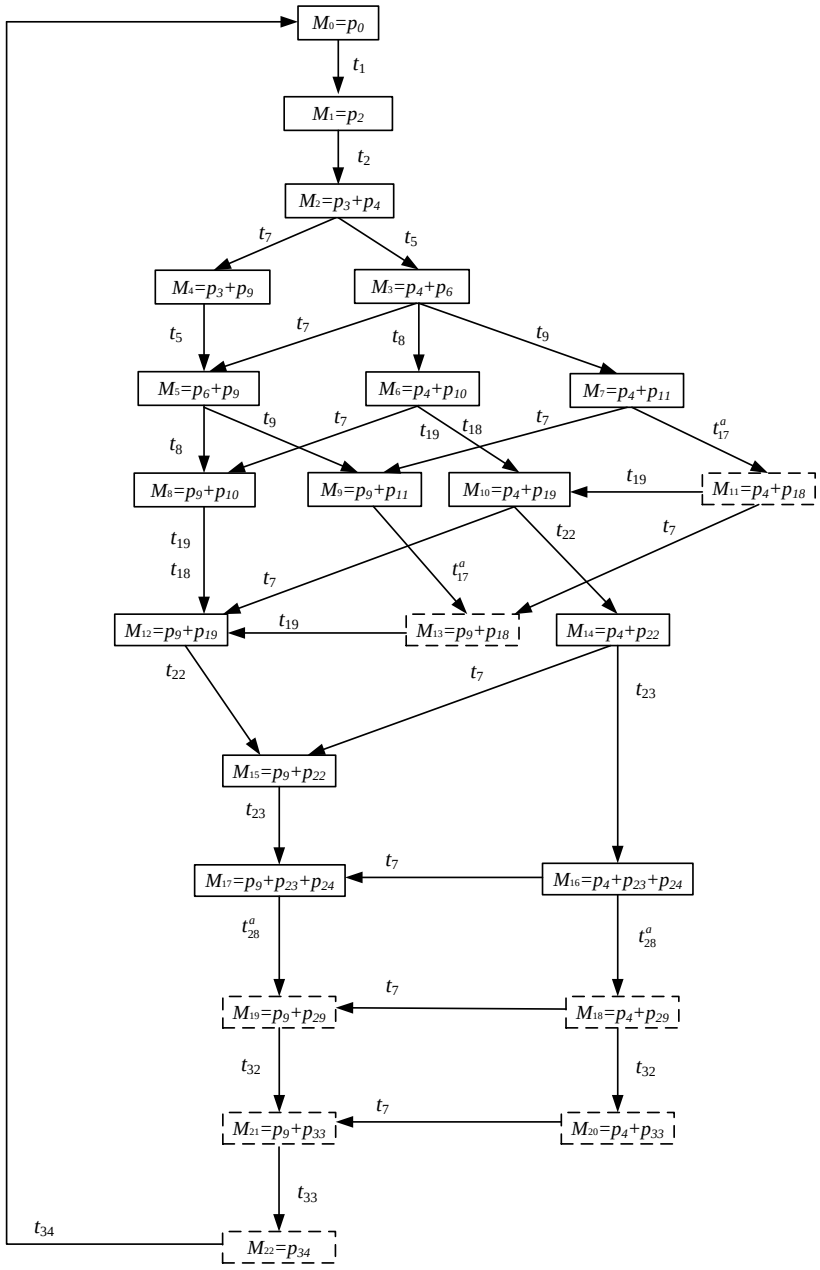

Figure 18. The basis attack model $G_{\mathcal{B}}$.

According to the safe controllability condition proposed in this article, we use Algorithms 3 and 4 to determine whether the plant satisfies AE-safe controllability. Part of the basis diagnoser G_D is shown in Figure 19. It can be seen that the plant can still arrive at state M_{19}. At this point, the system detects that an attack has occurred. Next, the system will continue to reach the state M_{21} via the event $t_{32} \in T_{uc}$. There is no controllable event to interrupt the process between the attacked event and the unsafe state, thus violating AE-safe controllability.

The detection of AE-attacks by constructing a basis diagnoser largely improves the efficiency. By using the traditional automaton approach in this example, we generate 123 states and 234 arc relations,the state compression rate is 81.3%, and the arc relation compression rate is 85.5%. Since the reachability graph is equivalent to a finite state automaton, the above comparison is based on the reachability graph.

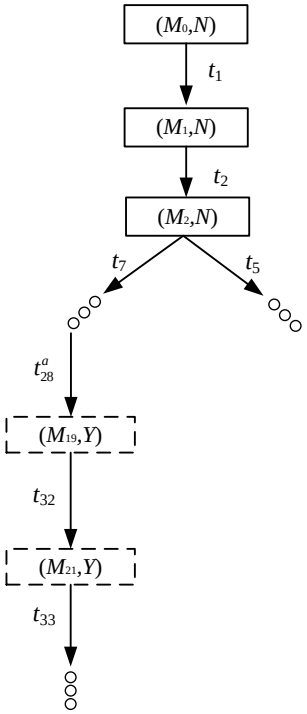

Figure 19. A part of the basis diagnoser G_D.

7. Conclusions

In this article, we studied the attack detection of AE-attacks in a supervisory control system. In a supervisory control system, actuator signals are vulnerable to manipulation by an attacker. An attacker will enable events that have been disabled by a supervisor in order to make the system reach unsafe states. We use the technique under Petri nets to develop attack detection methods to protect the system by disabling all controllable events after detecting an attack. First, we introduce a general framework for attack detection that models the plant as a Petri net to describe the system behaviors. Second, we simplify the attack model using basis markings to construct a basis attack model to analyze the system behaviors after an attack occurs. The basis attack model satisfies the properties of a closed-loop control system and reduces the number of states in the plant. Third, to prevent the attack from causing damage to the system, we also build the basis diagnoser to judge AE-safe controllability. After an attack is detected, the supervisor disables all controllable events to prevent the system from reaching an unsafe state. If successful, the system satisfies AE-safe controllability; otherwise, it does not. Compared with the traditional method, our method improves the detection efficiency and alleviates the state explosion problem. Finally, we also provide an offline solution to the attack detection problem. In this article, we consider AE-attacks to explain our framework and results. In fact, our method is general and available for other types of attacks. In future work, we will extend the approach to unbounded nets and relax the assumptions and we will also consider new types of attacks similar to the spread of viruses [36–38] in the framework of DESs.

Author Contributions: Conceptualization, Z.Y.; methodology, Z.Y. and X.C.; formal analysis, X.C.; investigation, X.D.; writing—original draft preparation, X.D.; writing—review and editing, X.C., X.L. and L.Z.; funding acquisition, Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grants 62273272 and 61873277, the Key Research and Development Program of Shaanxi Province under Grant 2023-YBGY-243, the Natural Science Foundation of Shannxi Province under

Grant 2022JQ-606, the Research Plan of Department of Education of Shaanxi Province under Grant 21JK0752, and the Youth Innovation Team of Shaanxi Universities.

Data Availability Statement: Not applicable.

Acknowledgments: The authors sincerely appreciate the editor and anonymous referees for their careful reading and helpful comments to improve this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lin, L.; Zhu, Y.; Su, R. Synthesis of covert actuator attackers for free. *Discret. Event Dyn. Syst.* **2020**, *30*, 561–577. [CrossRef]
2. Yu, Z.; Gao, H.; Wang, D.; Alnuaim, A.A.; Firdausi, M.; Mostafa, A.M. SEI^{2}RS malware propagation model considering two infection rates in cyber–physical systems. *Phys. A Stat. Mech. Appl.* **2022**, *597*, 127207. [CrossRef]
3. Meira-Góess, R.; Kang, E.; Kwong, R.H.; Lafortune, S. Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems. *Automatica* **2020**, *121*, 109172. [CrossRef]
4. Meira-Góess, R.; Kang, E.; Kwong, R.H.; Lafortune, S. Stealthy deception attacks for cyber–physical systems. In Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, VIC, Australia, 12–15 December 2017; pp. 4224–4230.
5. Zhang, D.; Wang, Q.G.; Feng, G.; Shi, Y.; Vasilakos, A.V. A survey on attack detection, estimation and control of industrial cyber–physical systems. *ISA Trans.* **2021**, *116*, 1–16. [CrossRef] [PubMed]
6. Yu, Z.; Sohail, A.; Jamil, M.; Beg, O.; Tavares, J.M.R. Hybrid algorithm for the classification of fractal designs and images. *Fractals* 2022, *accepted*. [CrossRef]
7. Hou, Y.; Shen, Y.; Li, Q.; Ji, Y.; Li, W. Modeling and optimal supervisory control of networked discrete-event systems and their application in traffic management. *Mathematics* **2023**, *11*, 3. [CrossRef]
8. Yu, Z.; Wang, H.; Wang, D.; Li, Z.; Song, H. CGFuzzer: A fuzzing approach based on coverage-guided generative adversarial networks for industrial IoT protocols. *IEEE Internet Things J.* **2022**, *9*, 21607–21619. [CrossRef]
9. Cong, X.; Fanti, M.P.; Mangini, A.M.; Li, Z. Critical observability of discrete-event systems in a Petri net framework. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 2789–2799. [CrossRef]
10. Thorsley, D.; Teneketzis, D. Intrusion detection in controlled discrete event systems. In Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, 13–15 December 2006; pp. 6047–6054.
11. Wakaiki, M.; Tabuada, P.; Hespanha, J.P. Supervisory control of discrete-event systems under attacks. *Dyn. Games Appl.* **2019**, *9*, 965–983. [CrossRef]
12. Wang, Y.; Pajic, M. Supervisory control of discrete event systems in the presence of sensor and actuator attacks. In Proceedings of the 2019 IEEE 58th Conference on Decision and Control (CDC), Nice, France, 11–13 December 2019; pp. 5350–5355.
13. You, D.; Wang, S.; Zhou, M.; Seatzu, C. Supervisory control of Petri nets in the presence of replacement attacks. *IEEE Trans. Autom. Control* **2021**, *67*, 1466–1473. [CrossRef]
14. You, D.; Wang, S.; Zhou, M.; Seatzu, C. Supervisor synthesis to thwart cyberattack with bounded sensor reading alterations. *Automatica* **2018**, *94*, 35–44.
15. Agarwal, M. Rogue twin attack detection: A discrete event system paradigm approach. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 1813–1818.
16. Fritz, R.; Zhang, P. Modeling and detection of cyberattacks on discrete event systems. *IFAC-PapersOnLine* **2018**, *51*, 285–290. [CrossRef]
17. Wang, Y.; Li, Y.; Yu, Z.; Wu, N.; Li, Z. Supervisory control of discrete-event systems under external attacks. *Inf. Sci.* **2021**, *562*, 398–413. [CrossRef]
18. Zhang, Q.; Seatzu, C.; Li, Z.; Giua, A. Stealthy sensor attacks for plants modeled by labeled Petri nets. *IFAC-PapersOnLine* **2020**, *53*, 14–20. [CrossRef]
19. Ma, Z.; Cai, K. On Resilient Supervisory Control Against Indefinite Actuator Attacks in Discrete-Event Systems. *IEEE Control Syst. Lett.* **2022**, *6*, 2942–2947. [CrossRef]
20. Yao, J.; Yin, X.; Li, S. On attack mitigation in supervisory control systems: A tolerant control approach. In Proceedings of the 2020 59th IEEE Conference on Decision and Control (CDC), Jeju, Republic of Korea, 14–18 December 2020; pp. 4504–4510.
21. Rashidinejad, A.; Wetzels, B.; Reniers, M.; Lin, L.; Zhu, Y.; Su, R. Supervisory control of discrete-event systems under attacks: An overview and outlook. In Proceedings of the 2019 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019; pp. 1732–1739.
22. Zheng, S.; Shu, S.; Lin, F. Modeling and control of discrete event systems under joint sensor-actuator cyberattacks. In Proceedings of the 2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE), Dalian, China, 15–17 July 2021; pp. 216–220.
23. Carvalho, L.K.; Wu, Y.C.; Kwong, R.; Lafortune, S. Detection and mitigation of classes of attacks in supervisory control systems. *Automatica* **2018**, *97*, 121–133. [CrossRef]

24. Cong, X.; Fanti, M.P.; Mangini, A.M.; Li, Z. Decentralized Diagnosis by Petri Nets and Integer Linear Programming. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *48*, 1689–1700. [CrossRef]
25. Ma, Z.; Tong, Y.; Li, Z.; Giua, A. Basis marking representation of Petri net reachability spaces and its application to the reachability problem. *IEEE Trans. Autom. Control* **2017**, *62*, 1078–1093. [CrossRef]
26. Cassandras, C.G.; Lafortune, S. *Introduction to Discrete Event Systems*, 3rd ed.; Springer: Cham, Switzerland, 2021.
27. Cabasino, M.P.; Giua, A.; Pocci, M.; Seatzu, C. Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems. *Control Eng. Pract.* **2011**, *19*, 989–1001. [CrossRef]
28. Wonham, W.M.; Cai, K. *Supervisory Control of Discrete-Event Systems*, 1st ed.; Springer: Cham, Switzerland, 2019.
29. Paoli, A.; Sartini, M.; Lafortune, S. Active fault tolerant control of discrete event systems using online diagnostics. *Automatica* **2011**, *47*, 639–649. [CrossRef]
30. Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; Teneketzis, D. Diagnosability of discrete-event systems. *IEEE Trans. Autom. Control* **1995**, *40*, 1555–1575. [CrossRef]
31. Moreira, M.V.; Jesus, T.C.; Basilio, J.C. Polynomial time verification of decentralized diagnosability of discrete event systems. *IEEE Trans. Autom. Control* **2011**, *56*, 1679–1684. [CrossRef]
32. Yoo, T.S.; Lafortune, S. Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Trans. Autom. Control* **2002**, *47*, 1491–1495.
33. Jiang, S.; Huang, Z.; Chandra, V.; Kumar, R. A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Trans. Autom. Control* **2001**, *46*, 1318–1321. [CrossRef]
34. Cong, X.; Fanti, M.P.; Mangini, A.M.; Li, Z. Critical observability of labeled time Petri net systems. *IEEE Trans. Automat. Sci. Eng.* 2022, *Early Access*. [CrossRef]
35. Zou, M.; Tong, Y.; Ma, Z. PNBA: A software for marking estimation and reconfiguration in Petri nets using basis marking analysis. *IFAC-PapersOnLine* **2022**, *55*, 180–187. [CrossRef]
36. Yu, Z.; Sohail, A.; Arif, R.; Nutini, A.; Nofal, T.A.; Tunc, S. Modeling the crossover behavior of the bacterial infection with the COVID-19 epidemics. *Results Phys.* **2022**, *39*, 105774. [CrossRef]
37. Yu, Z.; Sohail, A.; Nofal, T.A.; Tavares, J.M.R. Explainability of neural network clustering in interpreting the COVID-19 emergency data. *Fractals* **2022**, *30*, 2240122. [CrossRef]
38. Yu, Z.; Ellahi, R.; Nutini, A.; Sohail, A.; Sait, S.M. Modeling and simulations of COVID-19 molecular mechanism induced by cytokines storm during SARS-CoV2 infection. *J. Mol. Liquids* **2020**, *327*, 114863. [CrossRef]