A general real-time control approach of intrusion response for industrial automation systems

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Citation: HUANG, S. ... et al., 2016. A general real-time control approach of intrusion response for industrial automation systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 46 (8), pp. 1021 - 1035.

Additional Information:

• © IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Metadata Record: https://dspace.lboro.ac.uk/2134/25527

Version: Accepted for publication

Publisher: © IEEE

Please cite the published version.
A General Real-Time Control Approach of Intrusion Response for Industrial Automation Systems

Shuang Huang, Chunjie Zhou, Naixue Xiong, Senior Member, IEEE, Shuang-Hua Yang, Senior Member, IEEE, Yuanqing Qin, and Qi Zhang

Abstract—Intrusion response is a critical part of security protection. Compared with IT systems, industrial automation systems (IASs) have greater timeliness and availability demands. Real-time security policy enforcement of intrusion response is a challenge facing intrusion response for IASs. Inappropriate enforcement of the security policy can influence normal operation of the control system, and the loss caused by this security policy may even exceed that caused by cyberattacks. However, existing research about intrusion response focuses on security policy decisions and ignores security policy execution. This paper proposes a general, real-time control approach based on table-driven scheduling of intrusion response in IASs to address the problem of security policy execution. Security policy consists of a service group, with each type of service support by a realization task set. Realization tasks from several task sets can be combined to form a response task set. In the proposed approach, first, a response task set is generated by a nondominated sorting genetic algorithm (GA) II with joint consideration of security performance and cost. Then, the system is reconfigured through an integrated scheduling scheme where system tasks and response tasks are mapped and scheduled together based on a GA. Furthermore, results from both numerical simulations and a real-application simulation show that the proposed method can implement the security policy in time with little effect on the system.

Index Terms—Industrial automation, intrusion response, security protection, task scheduling.

I. INTRODUCTION

MODERN industrial automation systems (IASs) are composed of physical plants that perform the physical processes and networks of embedded computers that perform the computational processes necessary to monitor and control the physical processes [1]. IASs are extensively applied in national economical industrial fields, such as water treatment, chemical processing, oil and natural gas distribution, and smart grids, which are at the core of critical infrastructures [2]–[5]. With the wide application of information and communication technologies to IASs and the rapid growth of viruses in IASs [6], [7], improving security protection has become a critical task that system designers must face [1], [8]. Security protection of IASs emphasizes the concept of in-depth defense [9]. Security protection provided by field devices is the last defense of the in-depth defense; it provides intrinsic security for field control systems. Much research about security protection integrated control theory for field control systems has been presented [10], [11], and some literature has studied the security protection at the field control layer from the perspective of software implementation [12]–[14]. As the workflow of process industry automation is coherent, security protection of process industry automation should be done from a perspective of whole system. In addition to other security mechanisms such as industrial firewalls and intrusion detection and response are also two key elements of the dynamic security protection of IASs [6], [15], [16]. Research working in the field of security protection of IASs is focusing on intrusion detection technology [5], [17]–[20]. Comparatively, there are few studies about intrusion response of industrial automation [21]–[23].

Intrusion response systems (IRSs) can be classified into three types: 1) manual IRSs; 2) semiautomatic IRSs; and 3) automatic IRSs. IASs have high availability demands, which means the manual and the semi-automatic intrusion response will not meet the security protection requirements for IASs. Automatic intrusion response has been a research topic in the IT domain for several years. Foo et al. [24] presented an automated response mechanism in an intrusion tolerant system, called ADEPTS, which uses a graph of intrusion goals (I-GRAPH) as the underlying representation in the system. Zonouz et al. [21] proposed a new approach about automated intrusion response called RRE which can analyze undesired security events and their countermeasures based on attack-response trees. Fessi et al. [23] designed a multiattribute genetic algorithm (GA) approach for handling a multiattribute decision problem in intrusion response. These works focus on security policy decision and ignore the security policy execution, i.e., instant intrusion response.

In intrusion response, a security policy contains many different types of security services. Furthermore, in dynamic security protection, security policy changes in real-time [15].
For example, in a certain situation, a security policy generated by a security policy decision is composed of an encryption service and an access control service. When the attack situation changes, a security policy decision may generate a new security policy that is composed of access control services only. Each type of security service can be implemented by various realization tasks. For instance, the encryption service can be realized by many cryptographic algorithms, such as Rivest Shamir Adlemen (RSA), Rivest Cipher 4 (RC4), or data encrypt standard (DES). Therefore, the response task set generated from a security policy is also changing in real-time. On the other hand, system tasks of IASs have real-time characteristics. They are simple and fixed, and these tasks are generally scheduled by table-driven scheduling algorithms to achieve predictable behaviors [25], [26]. Thus, to execute a security policy in IASs, response tasks cannot be treated in a simple way (such as simply consider them as new, high priority tasks). Execution of the security policy without any countermeasures may affect the system task behaviors and the normal operation of the system [27]. For example, a security policy is composed of an encryption service that is realized by a task. If this task is executed without considering the execution of other tasks in the system, it may affect the real-time performance of these other tasks or make some tasks unable to be executed smoothly within the required time. The loss caused by inappropriate execution of a security policy may even be more than that caused by cyberattacks. IASs are quite different from IT systems in some ways. IASs have an inherent character of real-time response and determinacy that is embodied by real-time tasks and communication. In time-based security (TBS) theory, security is defined as a system that can detect and respond to cyberattacks before security threats happen or reach the target system [28], [29], i.e., timeliness is the key issue of online security protection.

Task scheduling is an important guarantee of effective task execution [30], [31]. There is some literature available to research task scheduling and optimization with multiconstraints [32]–[34]. Liu et al. [33] proposed a novel security constraint model based on the formulation of a scheduling problem for work-flow applications in distributed data-intensive computing environments. Saleh and Dong [34] designed a real-time multiagent design model and proposed an adaptive security-aware scheduling system that is combined with security service enhancement. These methods are designed for IT systems, and security requirements are regarded as constraints. They leave intrusion response tasks out of consideration. System tasks in IASs are generally simple and fixed, which are very different from the tasks in IT systems and are usually scheduled by table-driven scheduling algorithms [25], [26].

In this paper, we propose a general real-time control approach based on table-driven task scheduling of intrusion response in IASs. In this approach, the response tasks and system tasks are integrated into the task scheduling table with various requirements being considered to achieve instant intrusion response to cyberattacks. First, through a multiobjective optimization method based on the nondominated sorting GA II (NSGA-II), an intrusion response task set is generated from the security policy with consideration of security performance and cost in IASs. Second, a unified task model is conducted based on the combination of the intrusion response task set and a system task set. A distributed table-driven task scheduling and optimization scheme based on a GA and directed acyclic graph (DAG) is designed to update the task scheduling tables. Then, the system is reconfigured by updating the task scheduling tables to ensure the execution of intrusion response tasks and the normal operation of system functions. At last, the effectiveness of the proposed approach is verified by numerical simulations and application simulations based on a real-application system.

The rest of this paper is organized as follows. In Section II, the real-time issue of intrusion response in IASs is presented, and then a general real-time control architecture of intrusion response that contains response task set generation and integrated scheduling is proposed. Section III designs a multiobjective optimization method for response task set generation based on a given security policy. Integrated scheduling and optimization for the combination of the response task set and the system task set is presented by using a GA in Section IV. Sections V and VI show numerical simulations and application simulations, respectively. Finally, the study is concluded in Section VII with a summary of the research undertaken and plans for future research.

II. REAL-TIME CONTROL ISSUE OF INTRUSION RESPONSE FOR IASs

A. Problem Description

IASs are characterized by real-time and high reliability [35]. Meanwhile, IASs are industrial production systems and are safety critical systems. They should be able to defend themselves against the complicated and changing cyberattacks at runtime, i.e., they need to have dynamic security protection. The dynamic security protection for IASs is essential against cyberattacks [15]. Fig. 1 shows a general dynamic security protection model for IASs. Dynamic security protection of IASs consists of two parts: 1) intrusion detection and 2) intrusion response. Intrusion response is responsible for security policy decision and security policy enforcement. The security policy is generated at the security policy decision phase [22], while the security policy is interpreted and executed in the systems at the security policy enforcement phase.

Generally, a security policy is composed of different kinds of security services forming a security service
group (SSG) [36], and each type of security service can be implemented by various realization tasks. Thus, in the process of dynamic security protection, real-time security policies can be interpreted as different numbers and different types of response tasks that can take up system resources, such as communication bandwidth, CPU time, and memory. Therefore, the enforcement of these response tasks may disrupt the system task scheduling and affect the system’s normal operation. On the other hand, the response tasks generated from security policy directly determine the security performance of IRSs. Consequently, an integrated real-time task enforcement approach of intrusion response for IASs is needed to ensure the smooth and efficient enforcement of security policy and the system’s normal operation.

In this paper, we focus on the security policy enforcement component of intrusion response for IASs. A general real-time table-driven task scheduling approach of intrusion response is designed to solve this problem, where response tasks and system tasks are scheduled and optimized together with consideration of multiple constraints in IASs. The response tasks are generated dynamically according to a security policy, which are used to realize security services in the security policy. The system tasks include functional tasks and other nonfunctional (not pertaining to security) tasks.

B. Formalizing the Problem

An IAS is considered as a combination of a system task set $T^*_{sys}$ and a node set $N = \{p_i\}$. Suppose that the system tasks are dependent, nonpreemptible periodic tasks, and all nodes in the node set $N$ to be homogenous. IASs nodes include control devices, intelligent sensors, and intelligent actuators. Let $P$ represent a given security policy, and as mentioned above, the security policy can be regarded as an SSG composed of different kinds of security services [36]. So the security policy $P$ can be formalized as

$$P = \{S_1, \ldots, S_i, \ldots, S_k\}$$  \hspace{1cm} (1)

where $S_i$ is a security service and can be realized by many realization tasks, that is

$$S_i = R_i = \{r_{i1}, r_{i2}, \ldots, r_{iq}\}$$  \hspace{1cm} (2)

where $R_i$ is the realization task set of security service $S_i$. For example, the encryption service is a security service, which can be realized by many cryptographic algorithms, such as RSA, RC4, and DES. The parameters of a response task set $R_i$ are obtained offline. In this paper, we focus on security policy enforcement of IRSs based on a given security policy [shown as (1)], and do not discuss security policy decisions. Define $T^r$ is the response task set that is generated from security policy $P = \{S_1, \ldots, S_i, \ldots, S_k\}$, and the global task set $T$ is used to represent the combination of the response task set $T^r$ and the system task set $T^*_{sys}$. For easy of understanding, the notations and their meanings used in this paper are summarized in Table I.

![Fig. 2. Real-time control architecture of intrusion response for IASs.](image-url)

<table>
<thead>
<tr>
<th>Table I: Definitions of Notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notation</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>$P$</td>
</tr>
<tr>
<td>$S_i$</td>
</tr>
<tr>
<td>$R_i$</td>
</tr>
<tr>
<td>$r_{ij}$</td>
</tr>
<tr>
<td>$e_{ij}$</td>
</tr>
<tr>
<td>$w_{ij}$</td>
</tr>
<tr>
<td>$CC_{ij}$</td>
</tr>
<tr>
<td>$P_{ij}$</td>
</tr>
<tr>
<td>$T^r_i$</td>
</tr>
<tr>
<td>$k$</td>
</tr>
<tr>
<td>$PT_i$</td>
</tr>
<tr>
<td>$CC_i$</td>
</tr>
<tr>
<td>$HT_i$</td>
</tr>
<tr>
<td>$T^*_{sys}$</td>
</tr>
<tr>
<td>$T$</td>
</tr>
<tr>
<td>$n_i$</td>
</tr>
<tr>
<td>$MCT$</td>
</tr>
<tr>
<td>$\tau_i$</td>
</tr>
<tr>
<td>$w_{i}$</td>
</tr>
<tr>
<td>$st_{i}$</td>
</tr>
<tr>
<td>$pl_{i}$</td>
</tr>
<tr>
<td>$N$</td>
</tr>
<tr>
<td>$m$</td>
</tr>
<tr>
<td>$p_i$</td>
</tr>
<tr>
<td>$e_{i,j}$</td>
</tr>
<tr>
<td>$w(e_{i,j})$</td>
</tr>
</tbody>
</table>

C. Real-Time Control Architecture of Intrusion Response for IASs

To ensure enforcement of the intrusion response of IASs, a real-time control approach of intrusion response based on table-driven task scheduling and optimization is proposed, which is shown in Fig. 2.

As shown in Fig. 2, the intrusion detection and the security policy decision components are executed periodically, and the security policy generated from the security policy decision component is the real-time input of security policy enforcement component. This is a two-stage process containing response task set generation and integrated scheduling. In the response task set generation stage, with consideration of security performance requirements and time constraints, the Pareto solution of a response task set is generated from the security policy through the NSGA-II. Then, a solution in the form of a response task set having the highest security...
performance in the Pareto solution is selected as the optimal response task set. In the integrated scheduling stage, the response task set $T^*$ is first combined with system task set $T^{sys}$ to generate the global system task set $T$. Then, based on a DAG, a GA-based task scheduler is used to reconfigure the global system scheduling tables. If the global system task set is unschedulable, the response task set with slightly lower security performance will be re-elected as the optimizing response task set. The flowchart of the task scheduling and optimization is shown in Fig. 3.

III. RESPONSE TASK SET GENERATION OF INTRUSION RESPONSE FOR IASs

In this section, a response task set generation scheme of intrusion response is presented. We first model the problems of response task set generation, then we solve this problem using NSGA-II.

A. Modeling the Problem

The realization task $r_{ij}$ is defined as

$$r_{ij} = (c_{ij}, w_{ij}, cc_{ij}, p_{l_{ij}})$$

where $c_{ij}$ and $w_{ij}$ are the cycle time and (worst case) execution time of the realization task $r_{ij}$, respectively. $cc_{ij}$ is defined as the communication load of the realization task $r_{ij}$, which is represented as the byte count of the exchanged data of the realization task $r_{ij}$ in one period time $c_{ij}$. $p_{l_{ij}}$ is the security protection time that the realization task $r_{ij}$ can provide, which means the duration time that the attacker is able to achieve an attack goal in the system when the realization task $r_{ij}$ is employed [28], [29]. Thus, $p_{l_{ij}}$ of the $r_{ij}$ is different for the different types of attacks. The first three parameters of the $r_{ij}$ can be determined in advance through evaluation [31], [36], [37], while the $p_{l_{ij}}$ of the $r_{ij}$ for each type of attack can be listed online and determined in a security policy decision. It is important to note that heterogeneous processors are not considered here, i.e., all processors are treated the same.

Then, the response task set $T^r = \{r'_1, \ldots, r'_s, \ldots, r'_i\}$ is generated from the security policy $P = \{S_1, \ldots, S_i, \ldots, S_k\}$. Each individual response task $r'_i$ is chosen from the corresponding realization task set $R_i$. Thus, a security policy may have different security performances because a security policy can be realized by different SSGs that may have different security performances. For instance, the security performance of two cryptographic algorithms (e.g., RSA and DES) is obviously different. Security performance can be measured by time, i.e., TBS theory [28]. Security is defined as an intrusion attack that is detected and responded to in time [28], [29]. Thus, systems must be recovered before attackers achieve their intrusion goals. As shown in Fig. 4, security is formalized as follows:

$$PT > Det_r + Res_t$$

where $PT$ is the security protection time provided by the SSG, $Det_t$ and $Res_t$ are the intrusion detection time and the intrusion response time, respectively. The latter two parameters can be considered as constants for a specific system. Security time margin (STM) is defined as the difference between $PT$ and $Det_t + Res_t$, i.e., $STM = PT - (Det_t + Res_t)$. Thus, $STM \leq 0$ represents a system that is insecure, while $STM > 0$ represents a system that is secure. The STM can represent the security level.

For competition reasons, many intelligent devices at field control layer in IASs are implemented based on embedded systems, where the CPU performance, communication bandwidth, and memory size of IASs are designed according to rather limited requirements. Because of the business competition in IASs, cost is an important factor that must be considered during design and implementation. Most intelligent devices at the field control layer in IASs are implemented based on embedded systems, where the CPU performance, communication bandwidth, and memory size are designed according to rather limited requirements. Thus, in this paper, the IRS is constructed under certain resource constraints. In addition, the IRS is used to mitigate the effect of a cyberattack and protect system functions, which are implemented in IAS nodes. Thus, IAS nodes share the CPU resources and communication bus with the IRS, which requires that the response tasks of the IRS always affects the system’s normal operation. So, the response tasks of IRS should be considered together with system function tasks to mitigate impacts of the IRS on system functions.

Therefore, three optimization goals should be considered in the process of the response task set generation: security performance, the communication load and execution time of task. Let a decision variable $x_{ij}$ denote whether the $j$th realization task in realization task set $R_i$ is selected to implement the
ith security service $S_i$, and $X = (X_1, \ldots, X_i, \ldots, X_k)$, where $X_i = (x_{ij}, \ldots, x_{ij}, x_{ij})$.

$$x_{ij} = \begin{cases} 1, & r_{ij} \text{ is selected} \\ 0, & r_{ij} \text{ is not selected} \end{cases} \quad (4)$$

$x_{ij}$ should satisfy the following constraints: $\sum_{j=1}^{q_i} x_{ij} = 1$.

Then, the security protection time of the response task set $T' = \{t'_1, \ldots, t'_r, \ldots, t'_z\}$ can be calculated as $PT = \sum_{i=1}^{k} \sum_{j=1}^{q_i} x_{ij}p_{ij}$. The communication load per unit of time of the response task $r_{ij}$ is $cc_{ij}/cj_i$ and the communication load of the response task set can be represented as $CC = \sum_{i=1}^{k} \sum_{j=1}^{q_i} (x_{ij}cc_{ij}/cj_i)$. Analogously, the sum of the execution times of the response tasks in the response task set can be represented as $ET = \sum_{i=1}^{k} \sum_{j=1}^{q_i} (x_{ij}wt_{ij}/cj_i)$. Meanwhile, as shown in Fig. 4, to ensure that the system is secure, the following constraint must be satisfied: $PT \geq (Det_1 + Res_t)$, with $Det_1$ and $Res_t$ considered as constants for a specific system.

To generate the response task set $T'$ the problem is formalized as a multiobjective optimization problem (MOOP). The objective functions of this MOOP model are stated as

$$\begin{align*}
\text{Min ET} &= \sum_{i=1}^{k} \sum_{j=1}^{q_i} x_{ij}w_{ij}/cj_i \\
\text{Min CC} &= \sum_{i=1}^{k} \sum_{j=1}^{q_i} x_{ij}cc_{ij}/cj_i \\
\text{Max PT} &= \sum_{i=1}^{k} \sum_{j=1}^{q_i} x_{ij}pt_{ij}.
\end{align*} \quad (5)$$

And the constraints of the MOOP model are expressed as

$$\begin{align*}
PT &> (Det_1 + Res_t) \quad (6a) \\
\sum_{j=1}^{q_i} x_{ij} &= 1. \quad (6b)
\end{align*}$$

Meanwhile, the decision variable is represented as

$$X = (X_1, \ldots, X_i, \ldots, X_k) \quad (7)$$

where $X_i = (x_{ij}, \ldots, x_{ij}, x_{ij})$. Then, the problem of response task set generation can be formalized as (5) and (6).

B. Using the NSGA-II to Solve the MOOP Problem

Many evolutionary algorithms have been developed to solve MOOP problems, such as simulated annealing algorithm, particle swarm optimization algorithm, ant colony optimization algorithm, NSGA, and NSGA-II [38–43]. NSGA is proposed by Srinivas and Deb [39] to deal with MOOPs, where Goldberg’s nondomination criterion is used to determine solution ranks and fitness sharing is used to control the diversity of solutions in the search space. To improve the high-level sensitivity of NSGA to fitness sharing parameters, Deb et al. [44] introduced NSGA-II which includes a second-order sorting criterion called crowding distance and is faster and more reliable than NSGA. NSGA-II has been demonstrated to be among the most efficient algorithms for multiobjective optimization on a number of benchmark problems [44], as it has excellent search capability and flexibility [45]. Thus, we selected NSGA-II to solve the application problem of response task generation here.

NSGA II has good convergence toward the Pareto frontier without losing the diversity of the solutions, which introduces elitism into an evolutionary process and guarantees a diversity preserving mechanism. At the start of NSGA II, an initial population $P_0$ is randomly generated. Then, objective function values of the population $P_0$ are evaluated and the two criteria of each of the individual $i$'s in $P_0$, rank $i_{rank}$ and crowding distance $i_{distance}$, are calculated. After the initialization, the program enters a while loop. In the loop, there are six steps.

1) Selecting parent individuals from $P_t$ based on tournament selection strategy.
2) Generating offspring population $Q_t$ by crossover and mutation operators.
3) Evaluating objective function values of the population $Q_t$.
4) Merging $P_t$ and $Q_t$ to create $R_t$.
5) Calculating the two criteria ($i_{rank}$, $i_{distance}$) for each of the individual $i$ in $R_t$.
6) Selecting the best individuals in $R_t$ to generate $P_{t+1}$. These steps are repeated until the stopping condition is met. In the above steps, $P_t$ is the original population, $Q_t$ is the newly created population according to $P_t$, and $R_t$ is the combination of the populations of $P_t$ and $Q_t$. There are sophisticated algorithms to calculate the domination rank $i_{rank}$ and the crowding distance $i_{distance}$, such as described in [46], so that will not be covered again here.

Then, according to the MOOP problem shown as (5)–(7), the chromosome structure, genetic operations and optimal solution selection from the Pareto solution are designed as follows.

1) Chromosome Structure: The chromosome structure is determined by the decision variable in the problem. According to (7), each individual is coded as a binary vector $X$ of dimension $\sum_{i=1}^{k} q_i$ naturally, representing a decision vector.
2) Genetic Operations: Genetic operations are used to produce a new generation from an old generation. To effectively reduce the amount of the potential solutions, the crossover operator and the mutation operator are designed to meet the constraint shown as (6b).

1) Crossover Operator: In this research, the arithmetic crossover operator is used to produce offspring by swapping genetic parts of the parents. Thus, crossover point selection is key for the crossover operator. According to (6b), each $X_i$ should be regarded as an integral gene segment that cannot be divided. Therefore, the set of alternative crossover points is $\{q_1, q_1 + q_2, \ldots, \sum_{j=1}^{k} q_j, \ldots, \sum_{j=1}^{k} q_j\}$. The crossover point is randomly selected in the set, and the corresponding gene segments are exchanged to generate the offspring.
2) Mutation Operator: In the mutation operator process, a single bit in the parent string is flipped to create a new offspring string. However, to meet the constraint shown as (6b), first, a single bit randomly selected in the parent string is flipped, resulting in two cases.

a) Case 1: If the selected bit is “0,” then in the corresponding gene segment ($X_i$), the bit which
is “1” is changed to 0, and the selected bit is changed to 1.

b) Case 2: If the selected bit is 1, then the selected bit is changed to 0, and a bit randomly selected in the residual bits of the corresponding gene segment \(X_i\) is changed to 1.

3) Optimal Solution: The optimal solution for a response task set should be selected from the Pareto solution. In the process of multiobjective optimization, there are three optimization objectives: 1) security protection time; 2) communication load; and 3) execution time. In the Pareto solution, we select the optimal solution by making security protection time the highest priority. First, the solution with the largest security protection time in the Pareto solution is selected as the optimal solution. As shown in Fig. 2, after combining the response tasks with the system tasks, if the global system task set is unschedulable, the solution with the second largest security protection time in the Pareto solution is rescheduled as the optimal solution. The following section introduces the task schedule for the combination of the tasks.

IV. INTEGRATED SCHEDULING OF INTRUSION RESPONSE FOR IASs

Response task set \(T'^{\prime}\) will be executed with system task set \(T^{sys}\) after it is generated. In this section, an integrated scheduling scheme based on a DAG task model and GA is proposed for IASs. The architecture of the integrated scheduling is shown in Fig. 5. In the scheme initially, a global task set is first generated through unified modeling of the response tasks and system tasks. Then, global tasks are mapped and scheduled based on a DAG task model and GA, to achieve a scheduling table of each node of the system. During this process, communication tasks in nodes can be regarded as system tasks. Therefore, the communication scheduling table is also determined. Finally, the system is reconfigured by updating the scheduling tables to protect against cyberattacks.

A. System Model

As mentioned in Section II-B, an IAS is considered as a system task set \(T^{sys} = \{\tau_i\}\) and a node set \(N = \{pl_i\}; \ i = 1, \ldots, m\). The global task set \(T = \{\tau_i\}; \ i = 1, \ldots, n\) is a combination of the system task set \(T^{sys}\) and the response task set \(T', \ i.e., T = T' \cup T^{sys}\), where \(T' = \{t'_1, \ldots, t'_n\}\) and \(t'_i = (c_i, w_i, cci, pl_i)\).

Task \(\tau_i\) in the global task set is identified by the triples \(\tau_i = (w_i, st_i, pl_i)\), where \(w_i\) is the worst execution time of the task, \(st_i\) is the start time of the task (relative to the start time of the macro cycle), and \(pl_i\) is a list of the available node where the task can be assigned. Meanwhile, to model the global task set, a DAG is used to describe the dependent relationships among the global tasks. The global task set is represented by a binary group \(G = (T, E)\), where \(T\) is the global task set, and \(E = \{e_{ij}[i, j = 1, \ldots, n]\}\) is the set of directed arcs or edges between the tasks to represent the dependency. The \(w(e_{ij})\) represents the communication load (data exchanged from \(\tau_i\) to \(\tau_j\)) per one cycle time from \(\tau_i\) to \(\tau_j\). The edge \(e_{ij} \in E\) represents the dependent relationship that task \(\tau_i\) cannot start the execution of until task \(\tau_i\) completes its execution. Fig. 6 shows an example of the task model presented by DAG.

To facilitate constructing the DAG task model, the macro cycle time (MCT) is defined as the least common multiple of the cycle times of all the tasks in \(T\) and then all tasks can be converted into the tasks with the same cycle time. For example, assuming \(T' \cup T^{sys}\) be \(\{\tau_1, \tau_2, \tau_3\}\) and the cycle times of these tasks be 2, 4, and 4 ms, respectively, the MCT equals 4 ms, and \(\tau_1\) can be converted into two tasks whose cycle times are 4 ms. So, the parameter \(c_i\) in \(t'_i\) can be ignored. The parameter \(w_i\) in \(t'_i\) is the same as \(w_i\) in \(\tau_i\). The parameters \(cci_i\) and \(pl_i\) are not used in the integrated scheduling, so they can also be ignored. As the security services are usually not coupled with any hardware, the \(pl_i\) of \(t'_i\) equals \(N\).

Define the concept of the prodromal task set \(Pred(\tau_i)\) and successive task set \(Succ(\tau_i)\) as follows:

\[
\begin{align*}
Pred(\tau_i) &= \Gamma, \quad \text{if } e_{ij} \in E, \quad \tau_j \in \Gamma \\
Succ(\tau_i) &= \Phi, \quad \text{if } e_{ij} \in E, \quad \tau_j \in \Phi.
\end{align*}
\]  

(8)

If the prodromal task set is a null set, the corresponding task is called a start task; if the successive task set is null set, the corresponding task is called an end task. As shown in Fig. 6, \(\tau_1\) and \(\tau_2\) are the start tasks, while \(\tau_3\) and \(\tau_0\) are the end tasks. The prodromal task set of \(\tau_4\) is \(\{\tau_2, \tau_3\}\) and the successive task set of \(\tau_4\) is \(\{\tau_5, \tau_6\}\).

B. Task Mapping and Scheduling

As mentioned in Section IV-B, the global tasks are described by triples \(\tau_i = (w_i, st_i, pl_i)\) and the DAG model \(G\). In the process of the integrated task mapping scheduling, the following constraints should be taken into account.

1) Coupling Relationship Between Tasks: For a specific application in IASs, some tasks might need to run in the same node. Besides, as described in Section IV-A, the tasks which are converted from a same task must run...
in the same node. The coupling relationship between \( \tau_i \) and \( \tau_j \) can be represented as: if \( \tau_i \) runs in \( p_k \), \( \tau_j \) also runs in \( p_k \). In order to facilitate subsequent processing, we assume the coupled tasks are adjacent to each other.

2) **Dependent Relationship Between Tasks:** The dependent relationship between tasks is described by DAG, which can be expressed as: if \( e_{i,j} \in E \), \( s_{ij} \geq s_{ij} + w_{ij} \).

3) **Dependent Relationship Between Tasks and Nodes:** For a specific application in IASs, some tasks can only be executed by specific nodes. For instance, a data acquisition task must run in the corresponding sensor node. In task \( \tau_i = (w_{ij}, s_{ij}, p_{li}) \), \( p_{li} \) lists the nodes in which the task \( \tau_i \) can run.

4) **Communication Cost Constraint:** The weight of the edge in DAG \( w(e_{i,j}) \) is used to express the communication load from \( \tau_i \) to \( \tau_j \). In IASs, the communication bandwidth is limited, thus, we consider the communication load of the system being the sum of the communication load between nodes.

5) **Load Constraint:** In the process of task mapping, at least one task should be assigned to each node.

IASs belong to distributed systems, and communication load between nodes is an important index of the task mapping and scheduling in distributed systems [47]. In our research, we select the communication load as the optimization goal, and formulate the task mapping and scheduling as an optimization problem.

We used binary decision variable with two indices: \( Y = \{ y_{i,j} \}; \quad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, m \) to denote the tasks mapping on nodes, where \( y_{i,j} \) is defined as follows:

\[
y_{i,j} = \begin{cases} 
1, & \text{if } \tau_i \text{ is mapped on } p_j \\
0, & \text{otherwise}.
\end{cases}
\]  

(9)  

Because one task can only be mapped to one node, the \( y_{i,j} \) must satisfy the following constraint:

\[
\sum_{j=1}^{m} y_{i,j} = 1.
\]  

(10)  

Define a function \( Q(i,j) \) to denote whether two tasks are mapped on the same node

\[
Q(i,j) = \begin{cases} 
1; & \text{if } \tau_i \text{ and } \tau_j \text{ are mapped on the same node} \\
0; & \text{otherwise}.
\end{cases}
\]  

(11)  

And then, the communication load of the system can be calculated as follows:

\[
\text{Comm} = \sum_{e_{i,j} \in E} Q(i,j) \times w(e_{i,j}).
\]  

(12)  

To solve the task mapping and scheduling problem by a GA, the binary decision variables \( Y = \{ y_{i,j} \}; \quad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, m \) are used as the chromosome. The constraints 1) and 3) are satisfied by crossover and mutation operations, and the constraints 2), 4), and 5) are used to delete illegal individuals in the offspring population.

1) **Crossover Operation:** In the process of cross, \( Y_i = [y_{i,1}, \ldots, y_{i,j}, \ldots, y_{i,m}] \) is regarded as a meta-segment to meet (10). Then meta-segments are recombined through the single-point crossover to generate offspring, with the available crossover point set being \( \{1, 2, \ldots, n\} \). To meet constraint 1), numbers of the coupled tasks are removed in the available crossover point set. For example, if \( \tau_2 \)–\( \tau_4 \) are coupled, then, 2 and 3 are removed in the available crossover point set. Obviously, if the parents meet constraint 3), the offspring also meets constraint 3).

2) **Mutation Operation:** To meet (10), operation is similar to the mutation operation in Section III-B. To meet constraint 1), if \( \tau_i \) and \( \tau_j \) are coupled and \( Y_i \) mutate, then, let \( Y_i \) equal \( Y_i \) in each offspring.

After offspring population is generated by crossover and mutation operations, it should be checked by constraints 2), 4), and 5). For constraint 2), the dependent relationship between tasks is represented by DAG. Obviously, the completion of all the prodromal tasks of task \( \tau_1 \) is a necessary condition for task \( \tau_i \) to be started. Thus, tasks in the DAG are layered according to the dependent relationship in Algorithm 1. task_level is the number of task layers, and \( F_i \) is the task set to which the task belongs to the \( i \)th layer. Considering the whole DAG, the sum execution time (SET) that all \( n \) tasks execute on \( m \) nodes must not exceed the MCT. Then, for each individual represented by \( Y \), the SET can be calculated based on the layered tasks in Algorithm 2, and the start times \( ST \) of all tasks are also determined together. The \( \text{ST}(i) \) is used to record the available time of the \( i \)th node. In addition, it is easy to check each individual for constraints 4) and 5). After the optimal solution is obtained, the decision  

---

**Algorithm 1** Task Layering Algorithm Based on DAG in Integrated Scheduling  

**Require:** system task model DAG \( G \)  

**Ensure:** task_level \( F = \{ F_1, F_2, \ldots, F_{\text{task\_level}} \} \)  

1: \( F_1 = \{ \} \)  
2: for all \( \tau \) such that \( \tau \in T \) do  
3: \quad if \( \text{Pred}(\tau) = \text{NULL} \) then  
4: \quad \( F_1.\text{append}(\tau) \)  
5: \quad end if  
6: \quad end for  
7: while do  
8: \( F_{\text{imp}} = \{ \} \)  
9: for all \( \tau \) such that \( \tau \in F_i \) do  
10: \quad \( F_{\text{imp}} = F_{\text{imp}} \cup \text{Succ}(\tau) \)  
11: \quad end for  
12: if \( F_{\text{imp}} = \text{NULL} \) then  
13: \quad Break  
14: \quad else  
15: \quad \( F_{i+1} = F_{\text{imp}} \)  
16: \quad end if  
17: \quad end while  
18: task_level = \( i + 1 \)  
19: for \( i = 1 \) to task_level - 1 do  
20: \quad \( F_i = \{ e_{j,\text{pred}(i)->F_j} \} \)  
21: \quad end for  
22: return task_level \( F = \{ F_1, F_2, \ldots, F_{\text{task\_level}} \} \)
Algorithm 2 Task Scheduling Algorithm in Integrated Scheduling

Require: \( Y F = \{F_1, F_2, \ldots, F_{\text{task\_level}}\} \)
Ensure: \( ST = \{s_{t1}, s_{t2}, \ldots, s_{tn}\} \) \( SET \)
1: \[nt(1), nt(2), \ldots, nt(m)\] = \([0, 0, \ldots, 0]\)
2: for \( i = 1 \) to \( \text{task\_level} - 1 \) do
3: \( M_i = N_i = \{\} \)
4: for all \( \tau \) such that \( \tau \in F_i \) do
5: if \( \text{Succ}(\tau_k) = \text{NULL} \) then
6: \( N_i.append(\tau_k) \)
7: else
8: \( M_i.append(\tau_k) \)
9: end if
10: end for
11: sort(M_i(\tau, w))
12: sort(N_i(\tau, w))
13: for \( j = 1 \) to \text{len}(M_i) \) do
14: \( \tau = M_i(\tau_j) \)
15: \( \tau.st = \max_{\tau_j \in \text{Pred}(\tau)} (nt(P(\tau)), (\tau_k.st + \tau_k.w_k)) \)
16: \( nt(P(\tau)) = \tau.st + \tau.w \)
17: end for
18: for \( j = 1 \) to \text{len}(N_i) \) do
19: \( \tau = N_i(\tau_j) \)
20: \( \tau.st = \max_{\tau_j \in \text{Pred}(\tau)} (nt(P(\tau)), (\tau_k.st + \tau_k.w_k)) \)
21: \( nt(P(\tau)) = \tau.st + \tau.w \)
22: end for
23: end for
24: \( N = \bigcup_{i=1}^{n} N_i \)
25: \( SET = \max_{\tau \in N} (\tau_k.st + \tau_k.w_k) \)
26: return \( SET \) \( ST \)

V. NUMERICAL SIMULATIONS

In this section, the proposed method is implemented based on C++, and these simulations are carried out on a 2.6-GHz Pentium 4 laptop with 2 GB RAM to analyze and evaluate the performance and efficiency of the proposed real-time intrusion response approach in IASs. Four numerical scenarios are shown in Table II. As described in Section II-A, the number of response tasks \( k \) represents the dimension of the security policy. Table III shows the parameters predetermined of the corresponding realization task sets for the simulation. We suppose that the security policies are \( P_{\text{Scenario}1} = \{S_1, S_2, S_3, S_4\} \), \( P_{\text{Scenario}2} = \{S_1, S_2, S_3, S_4, S_5\} \), \( P_{\text{Scenario}3} = \{S_1, S_2, S_3, S_4\} \), and \( P_{\text{Scenario}4} = \{S_1, S_2, S_3, S_4, S_5\} \) for the four scenarios, respectively. Table IV shows the parameters of the system task set for the simulation. In scenario 1, the system task set is \( \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6\} \); in scenario 2, the system task set is \( \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\} \); in scenario 3, the system task set is \( \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8, \tau_9, \tau_{10}\} \); and in scenario 4, all tasks are used and tasks \( \tau_{11} - \tau_{10} \) are used twice. The DAGs are randomly generated based on the corresponding task sets.

A. Analysis of the Response Tasks Generation

During the process of the response tasks generation, the Pareto solution of the response task set is generated through NSGA II, and then the optimal solution of the response task set is selected according to the principle described in Section III-B. There are two main performance metrics for response task generation based on multiobjective optimization: 1) convergence to the Pareto optimal front and 2) the maintenance of a diverse set of solutions. Fig. 7(a) shows the Pareto solution fronts of response task generation in scenarios 1 and 2 at the 50th simulation step, and the relations between each pair of the three optimization objectives of the Pareto optimal are presented in Fig. 7(b)-(d). They show the tradeoff among security protection time, sum of communication loads of the response tasks, and sum of execution times of the response tasks. After selecting the optimal solution in the Pareto solutions, the result for the response task set generation in scenario 1 is \( T^*_1 = \{r_{11}, r_{22}, r_{32}, r_{43}\} \), and the results for the response task set generation in scenario 2 is \( T^*_2 = \{r_{11}, r_{22}, r_{32}, r_{43}, r_{52}\} \). The optimization solutions of the response task set generation are associated with the given security policy and the realization task sets, but not with the system task set and system nodes. Thus, Pareto optimal fronts of scenarios 2–4 are the same. Based on the above results, it can be seen that the Pareto solution of the response tasks generation are convergent and the Pareto solution fronts uniformly scattered.

B. Analysis of the Integrated Scheduling

The global system tasks (including response tasks and system tasks) are mapped and scheduled synthetically through GA based on the DAG model. Fig. 8 shows the variation of the fitness (communication load) for four scenarios during the genetic computation. It is found that all the individuals have reached almost the same fitness value after about ten generations for the four scenarios, which indicates that GA has reached the optimal solution after about ten generations. In Fig. 8, we can infer that the appropriate generation number of GA in the integrated scheduling module depends not only on the number of tasks but also on the number of the nodes in the system. This is mainly because the number of gene bits is determined by the product of the task number and the node number.
TABLE III
REALIZATION TASK SETS FOR SIMULATIONS

<table>
<thead>
<tr>
<th>Task</th>
<th>( c_{11} ) (ms)</th>
<th>( w_{11} ) (ms)</th>
<th>( c_{12} ) (bit)</th>
<th>( p_{11} ) (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_{11} )</td>
<td>1000</td>
<td>105</td>
<td>256</td>
<td>2500</td>
</tr>
<tr>
<td>( r_{12} )</td>
<td>1000</td>
<td>65</td>
<td>88</td>
<td>1850</td>
</tr>
<tr>
<td>( r_{13} )</td>
<td>1000</td>
<td>95</td>
<td>128</td>
<td>2300</td>
</tr>
<tr>
<td>( r_{14} )</td>
<td>1000</td>
<td>75</td>
<td>96</td>
<td>2100</td>
</tr>
<tr>
<td>( r_{15} )</td>
<td>1000</td>
<td>43</td>
<td>96</td>
<td>136</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>( c_{21} ) (ms)</th>
<th>( w_{21} ) (ms)</th>
<th>( c_{22} ) (bit)</th>
<th>( p_{21} ) (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_{21} )</td>
<td>1000</td>
<td>150</td>
<td>136</td>
<td>1000</td>
</tr>
<tr>
<td>( r_{22} )</td>
<td>1000</td>
<td>220</td>
<td>192</td>
<td>1500</td>
</tr>
<tr>
<td>( r_{23} )</td>
<td>1000</td>
<td>45</td>
<td>160</td>
<td>1100</td>
</tr>
<tr>
<td>( r_{24} )</td>
<td>1000</td>
<td>45</td>
<td>176</td>
<td>1300</td>
</tr>
<tr>
<td>( r_{25} )</td>
<td>1000</td>
<td>65</td>
<td>80</td>
<td>400</td>
</tr>
<tr>
<td>( r_{26} )</td>
<td>1000</td>
<td>75</td>
<td>112</td>
<td>500</td>
</tr>
<tr>
<td>( r_{27} )</td>
<td>1000</td>
<td>69</td>
<td>96</td>
<td>450</td>
</tr>
<tr>
<td>( r_{28} )</td>
<td>1000</td>
<td>80</td>
<td>136</td>
<td>510</td>
</tr>
</tbody>
</table>

Table V shows the results of the task mapping and scheduling for the four scenarios when set the related parameters are set as follows: initial_population = 100 and num_generation = 20. The results show that all tasks can be completed in one MCT, which means that all tasks are schedulable satisfying all constraints, and the execution of the response tasks has no effect on the system tasks. This demonstrates that the proposed method can achieve instant intrusion response.

C. Real-Time Performance Evaluation

The proposed approach aims to guarantee the smooth online enforcement of a security policy. On the other hand, the intrusion response that includes the proposed approach is also implemented as a task, so the execution time needs to be evaluated. Thus, we must consider the real-time performance of the proposed approach. To evaluate the real-time performance, 130 repeated tests for the four scenarios are carried out. Fig. 9 shows the real-time performance of the proposed approach of intrusion response based on task scheduling and optimization for the four scenarios in the repeated tests. It can be seen that the execution times of the proposed method for all four scenarios are less than 60 ms. The mean values are 14.82, 26.50, 26.58, and 46.28 ms, respectively. This is because the number of the gene bits in the response task set generation based on NSGA is \( \sum_{k=1}^{m} q_k \) and the number of the gene bits in the integrated scheduling based on GA is \( m \times n \). These two parameters are the important factors influencing the execution times of the proposed method. These results show that the proposed algorithm can be executed in a short time. It also means that a system can respond quickly to any cyber attacks.

VI. SIMULATION—SIMPLIFIED TEP CONTROL SYSTEM

To further verify the effects and feasibility of the proposed approach of intrusion response in real-application systems, in this section, we construct a simulation environment based on OPEN and MATLAB, and a simplified Tennessee Eastman process (TEP) control system is used to carry out the simulations.

A. Simulation Setup

1) Simplified TEP and Simulation System: We used a simplified TEP control system as Ricker [48] described to carry out the simulations. The simplified TEP control system is a chemical reactor plant and has been previously used in [49] and [50]. In the chemical process, two reactants, A and C, are used to generate the product D, while B is an inert product. The schematic of the simplified TEP control system is shown in Fig. 10. More details on the simplified TEP can be found in [49] and [50].

In the simulation, we focused on a field control system for industrial automation and do not consider links to the outside networks. The simplified TEP control system in an OPNET environment is shown in Fig. 11. The physical process is simulated in the MATLAB environment, and the control component is implemented in an OPNET environment. This simplified TEP model and the associated multiloop PI control law proposed in [48] are used. The communication network is a controller area network (CAN) bus, which is also implemented in the OPNET environment. The model of the physical process is called an agent node (PA in Fig. 11) in OPNET. The simulation system contains one master node (MN) and five slave nodes (one plant agent node and four controller nodes). All nodes are connected to a CAN bus to exchange messages.

Based on the above simulation system, the dynamic security protection shown in Fig. 1 is implemented. It is important to note that this paper focuses on security policy enforcement, thus the rest of the dynamic security protection is only realized in a simple way. An intrusion detection module is used...
to generate a security policy online that is used for the input to the security policy enforcement. The intrusion detection module can generate an integer that ranges between 0 and 3. 0 represents no cyberattack, while “1,” “2,” and “3” represent three types of cyberattacks. The intrusion detection module is implemented in all the nodes and used to simulate the intrusion detection component of Fig. 1. The intrusion detection module contains one task, i.e., intrusion detection, and the intrusion response module contains only one task, i.e., intrusion response. The CAN_TX and CAN_RX processes are responsible for the network communication for the node. The other tasks are in the APP process. Because we only consider the execution time of intrusion detection in this paper, in other nodes (slave nodes), a useless task is used to simulate the real-intrusion detection in the DETECT process. The APP, CAN_TX, and CAN_RX processes are these processes in the MN. As shown in Fig. 2, the updated scheduling tables are broadcasted through the CAN bus after they are changed. All nodes are reconfigured according to the corresponding scheduling tables.

2) Parameter Configuration: Table VI shows the system tasks of the simplified TEP control system. The execution times of these tasks are obtained by estimation, according to the corresponding code. The periods \( c_i \) and the \( pl_i \) are determined by the application. In the intrusion detection task, some dummy code is inserted to simulate the execution time of the real-intrusion detection. In practice, the PA node is non-existent, as it is only for communication between the physical system and the control system. Thus, the PA node and its tasks are not considered in the following simulations. To facilitate understanding, we label the nodes MN, FC, PC1, PC2, and CC for the simulation system shown in Fig. 11 as \( p_1-p_5 \), respectively.

Table VI. The proposed method of security policy enforcement is also implemented according to the descriptions in Sections III and IV. The process models of the MN node and the other nodes are presented in Fig. 12. In the MN, the DETECT process contains one task, i.e., intrusion detection, and the RESPONSE process also contains only one task, i.e., intrusion response. The CAN_TX and CAN_RX processes are responsible for the network communication for the node. The other tasks are in the APP process. Because we only consider the execution time of intrusion detection in this paper, in other nodes (slave nodes), a useless task is used to simulate the real-intrusion detection in the DETECT process. The APP, CAN_TX, and CAN_RX processes are these processes in the MN. As shown in Fig. 2, the updated scheduling tables are broadcasted through the CAN bus after they are changed. All nodes are reconfigured according to the corresponding scheduling tables.

Fig. 7. Pareto solutions of the MOOP for the response task set generation in the scenarios. (a) Pareto optimal fronts in scenarios 1 and 2. Relation between (b) ET and CC, (c) CC and PT, and (d) ET and PT.

Fig. 8. Convergence of the proposed integrated scheduling-based GA for the four scenarios.

Fig. 9. Real-time performance for four scenarios.
### Table V
Result of Task Mapping and Scheduling for the Four Scenarios

<table>
<thead>
<tr>
<th>Scenario 1</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Task</th>
<th>Start time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>$T_1$</td>
<td>$T_2$</td>
<td>$T_3$</td>
<td>$T_4$</td>
<td>$T_5$</td>
<td>$T_6$</td>
</tr>
<tr>
<td>Start time</td>
<td>0</td>
<td>220</td>
<td>230</td>
<td>350</td>
<td>0</td>
<td>390</td>
</tr>
</tbody>
</table>

Table VI
Parameters of System Tasks in the Simplified TEP Control System

<table>
<thead>
<tr>
<th>Task</th>
<th>$v_j$ (ms)</th>
<th>$c_j$ (ms)</th>
<th>$p_j$</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrusion detection</td>
<td>200</td>
<td>3000</td>
<td>$p_1$</td>
<td>$\tau_1$</td>
</tr>
<tr>
<td>Intrusion response</td>
<td>420</td>
<td>3000</td>
<td>$p_2$</td>
<td>$\tau_2$</td>
</tr>
<tr>
<td>CAN communication</td>
<td>110</td>
<td>3000</td>
<td>$p_3$</td>
<td>$\tau_3$</td>
</tr>
<tr>
<td>Pressure acquisition</td>
<td>50</td>
<td>1000</td>
<td>$p_4$, $p_5$</td>
<td>$\tau_4$, $\tau_5$, $\tau_6$</td>
</tr>
<tr>
<td>$u_1$ calculation</td>
<td>100</td>
<td>3000</td>
<td>$p_6$, $p_7$, $p_8$, $p_9$</td>
<td>$\tau_7$</td>
</tr>
<tr>
<td>Intrusion detection</td>
<td>120</td>
<td>3000</td>
<td>$p_1$</td>
<td>$\tau_8$</td>
</tr>
<tr>
<td>CAN communication</td>
<td>80</td>
<td>3000</td>
<td>$p_2$</td>
<td>$\tau_9$</td>
</tr>
<tr>
<td>Product rate acquisition</td>
<td>60</td>
<td>1000</td>
<td>$p_3$, $p_4$, $p_5$</td>
<td>$\tau_{10}$, $\tau_{11}$, $\tau_{12}$</td>
</tr>
<tr>
<td>Data processing</td>
<td>100</td>
<td>3000</td>
<td>$p_6$, $p_7$, $p_8$, $p_9$</td>
<td>$\tau_{13}$</td>
</tr>
<tr>
<td>Intrusion detection</td>
<td>120</td>
<td>3000</td>
<td>$p_1$</td>
<td>$\tau_{14}$</td>
</tr>
<tr>
<td>$u_2$ calculation</td>
<td>110</td>
<td>3000</td>
<td>$p_6$, $p_7$, $p_8$, $p_9$</td>
<td>$\tau_{15}$</td>
</tr>
<tr>
<td>CAN communication</td>
<td>80</td>
<td>3000</td>
<td>$p_2$</td>
<td>$\tau_{16}$</td>
</tr>
<tr>
<td>Pressure control</td>
<td>100</td>
<td>3000</td>
<td>$p_3$, $p_4$</td>
<td>$\tau_{17}$</td>
</tr>
<tr>
<td>Intrusion detection</td>
<td>120</td>
<td>3000</td>
<td>$p_1$</td>
<td>$\tau_{18}$</td>
</tr>
<tr>
<td>CAN communication</td>
<td>80</td>
<td>3000</td>
<td>$p_2$</td>
<td>$\tau_{19}$</td>
</tr>
<tr>
<td>Level acquisition</td>
<td>50</td>
<td>1000</td>
<td>$p_3$</td>
<td>$\tau_{20}$, $\tau_{21}$, $\tau_{22}$</td>
</tr>
<tr>
<td>CAN communication</td>
<td>80</td>
<td>3000</td>
<td>$p_1$</td>
<td>$\tau_{23}$</td>
</tr>
<tr>
<td>$u_3$ calculation</td>
<td>90</td>
<td>3000</td>
<td>$p_3$, $p_4$, $p_5$</td>
<td>$\tau_{24}$</td>
</tr>
<tr>
<td>Intrusion detection</td>
<td>120</td>
<td>3000</td>
<td>$p_1$</td>
<td>$\tau_{25}$</td>
</tr>
</tbody>
</table>

Fig. 10. Schematic of the simplified TEP control system.

Network: example-scenario2 [Subnet: top. Office Network]

Fig. 11. Simplified TEP control system in OPNET environment.

The decision table for the security policy is designed as follows: when there is no attack, the security policy $P$ is an empty set; when an attack of Type 1 occurs, the security policy is $P_{Type1} = \{S_1, S_2\}$; when an attack of Type 2 occurs, the security policy is $P_{Type2} = \{S_1, S_3\}$; for a Type 3 attack, the security policy is $P_{Type3} = \{S_1, S_2, S_3\}$. This means that the response tasks are generated from the corresponding security policy $P$. The security services $S_1$–$S_3$ that are used in the security policy can be implemented by the corresponding realization task sets $R_1$–$R_3$, respectively. The parameters of the realization tasks are given in Table VII. These realization tasks are prestored as functions in the nodes.

### B. Simulation Results

To verify the effectiveness of the proposed method in the simulation system, the intrusion detection component generates the following output:

$$\text{Output} = \begin{cases} 
0, & t \in [0, 100) \cup [70, 100) \\
1, & t \in [10, 30) \\
2, & t \in [30, 50) \\
3, & t \in [50, 70) 
\end{cases}$$

Table VIII shows the Pareto solution for the response task set generation for $t = 50$. As described in Section III-B, the...
optimal response task set is the one with the largest security protection time in the Pareto solution. Thus, the optimal solution of the response task set generation is $T^* = \{r_{11}, r_{23}, r_{32}\}$.

Table IX shows the execution of all the tasks from the perspective of task scheduling. Control performance is also an important factor for IASs. From the perspective of control engineering, it should also be guaranteed that the execution of a security policy will have no effect on the control performance of the system. Fig. 13 provides a comparison of dynamic response $y_F$ between the normal simplified TEP control system and the system with dynamic security protection. It clearly shows that the dynamic security protection has almost no influence on the dynamic response of the simplified TEP control system.

Then, through the integrated scheduling module, we can obtain the optimal solution of the integrated task mapping and the corresponding task scheduling table. Due to limited space, Table IX only shows the integrated task mapping and the corresponding task scheduling table for tasks reconfigured for $t = 50$. The result shows that all tasks (including system tasks and response tasks) are schedulable, and the sum of the execution times of all the nodes does not exceed the MCT (3000 ms). It also means that response tasks can successfully be carried out with no effect on the execution of the system tasks.

Table IX shows the execution of all the tasks from the perspective of task scheduling. Control performance is also an important factor for IASs. From the perspective of control engineering, it should also be guaranteed that the execution of a security policy will have no effect on the control performance of the system. Fig. 13 provides a comparison of dynamic response $y_F$ between the normal simplified TEP control system and the system with dynamic security protection. It clearly shows that the dynamic security protection has almost no influence on the dynamic response of the simplified TEP control system.
In the above simulations, there are three different security policies. This is because the proposed approach of intrusion response can be clearly explained based on these configurations, and security policy decision is not the focus of this paper. In real-time security protection of IASs, security policies generated from the security policy decision are complex and varied. From the above simulation results, it can be seen that the proposed real-time control approach of intrusion response is an effective method to guarantee the smooth, timely execution of the security policy without effect on system control performance.

VII. Conclusion

Timely intrusion detection and response is the key idea for online security protection for IASs. In this paper, we proposed a general real-time control approach based on table-driven task scheduling of intrusion response for IASs, which utilizes TBS criteria and integrates the response tasks and system tasks into the scheduling and optimization after consideration of a number of requirements. First, to meet the wide variety of security requirements for security protection of IASs, the security policy is formalized as a group of security services with different types. Each type of security service can be realized by many tasks, and these compose the realization task set for the corresponding security service. The response tasks are generated through an NSGA-II algorithm according to the security policy. Then, to mitigate the execution of the response tasks, an integrated scheduling strategy based on the GA is designed to map and schedule the system tasks and response tasks together.

The proposed method has been implemented using the C++ language and analyzed on several synthetic applications. Then, a simplified TEP control system was used to further verify the effectiveness of the proposed method in dynamic security protection based on OPNET and MATLAB environments. Simulation results have shown that the given security policy can be smoothly implemented and achieve the instant intrusion response without effect on the normal operation of the system. Currently, all nodes are considered as homogeneous. In fact, there are also many heterogeneous nodes in most IASs. Further improvement to this paper can be made by considering the inclusion of heterogeneous node. In addition, this paper focused on security policy enforcement, and did not discuss security policy generation. Therefore, security policy generation based on decision theory of intrusion response will be considered.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their helpful comments and suggestions for improving the quality of this paper.

References


Shuang Huang was born in 1986 in Hubei, China. He received the B.S. degree in automation from the Huazhong University of Science and Technology, Wuhan, China, in 2009, where he is currently pursuing the Ph.D. degree in control science and control engineering with the School of Automation.

His current research interests include industrial communication and industrial control systems with special focus on security.

Chunjie Zhou was born in Hubei, China, in 1965. He received the B.S., M.S., and Ph.D. degrees in control theory and control engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1988, 1991, and 2001, respectively.

He is currently a Doctoral Tutor Professor with the School of Automation, Huazhong University of Science and Technology. His current research interests include industrial communication, artificial intelligence, theory and application of networked control systems, and security protection of industrial control systems.
Naixue Xiong (M'08–SM'12) received the Ph.D. degrees in software engineering and dependable networks from Wuhan University, Wuhan, China, and the Japan Advanced Institute of Science and Technology, Nomi, Japan.

He was with the Wentworth Technology Institution, Georgia State University, Atlanta, GA, USA, for several years. He is currently a Full Professor with the School of Computer Science, Colorado Technical University, Colorado Springs, CO, USA. His current research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory.

Prof. Xiong was the General Chair, the Program Chair, the Publicity Chair, a PC Member, and an OC Member of over 100 international conferences, and a Reviewer of about 100 international journals, including the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. He serves as the Editor-in-Chief and an Associate Editor or an Editor Member for over ten international journals, including an Associate Editor for the IEEE TRANSACTION ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, the Editor-in-Chief for the Journal of Parallel and Cloud Computing, and a Guest Editor for over ten international journals, including the Sensor Journal, WINET, and MONET.

Shuang-Hua Yang (M’05–SM’06) received the B.S. degree in instrument and automation and the M.S. degree in process control from the University of Petroleum, Shandong, China, in 1983 and 1986, respectively, and the Ph.D. degree in intelligent systems from Zhejiang University, Hangzhou, China, in 1991.

He is currently a Professor of Networks and Control and the Director of the Networks, Communications and Control Systems Research Division, Loughborough University, Leicestershire, U.K. His current research interests include wireless sensor networks, networked control, safety critical-systems, and real-time software maintenance.

Prof. Yang is an Associate Editor of the International Journal of Systems Science, the International Journal of Automation and Computing, and the Arabian Journal for Science and Technology. He is an Editorial Advisory Board Member of the International Journal of Information and Computer Security and the Journal of Measurement and Control. He is a Fellow of the Institute of Measurement and Control and a Chartered Engineer in the U.K.

Qi Zhang was born in 1987 in Liaoning, China. He received the B.S. degree in automation from the Huazhong University of Science and Technology, Wuhan, China, in 2011, where he is currently pursuing the Ph.D. degree in control theory and control engineering with the School of Automation under the supervision of C. Zhou.

His current research interests include distributed systems/networks and industrial Ethernet.
A General Real-Time Control Approach of Intrusion Response for Industrial Automation Systems

Shuang Huang, Chunjie Zhou, Naixue Xiong, Senior Member, IEEE, Shuang-Hua Yang, Senior Member, IEEE, Yuanqing Qin, and Qi Zhang

Abstract—Intrusion response is a critical part of security protection. Compared with IT systems, industrial automation systems (IASs) have greater timeliness and availability demands. Real-time security policy enforcement of intrusion response is a challenge facing intrusion response for IASs. Inappropriate enforcement of the security policy can influence normal operation of the control system, and the loss caused by this security policy may even exceed that caused by cyberattacks. However, existing research about intrusion response focuses on security policy decisions and ignores security policy execution. This paper proposes a general, real-time control approach based on table-driven scheduling of intrusion response in IASs to address the problem of security policy execution. Security policy consists of a security service group, with each type of security service supported by a realization task set. Realization tasks from several task sets can be combined to form a response task set. In the proposed approach, first, a response task set is generated by a nondominated sorting genetic algorithm (GA) II with joint consideration of security performance and cost. Then, the system is reconfigured through an integrated scheduling scheme where system tasks and response tasks are mapped and scheduled together based on a GA. Furthermore, results from both numerical simulations and a real-application simulation show that the proposed method can implement the security policy in time with little effect on the system.

Index Terms—Industrial automation, intrusion response, security protection, task scheduling.

I. INTRODUCTION

MODERN industrial automation systems (IASs) are composed of physical plants that perform the physical processes and networks of embedded computers that perform the computational processes necessary to monitor and control the physical processes [1]. IASs are extensively applied in national economical industrial fields, such as water treatment, chemical processing, oil and natural gas distribution, and smart grids, which are at the core of critical infrastructures [2]–[5]. With the wide application of information and communication technologies to IASs and the rapid growth of viruses in IASPs [6], [7], improving security protection has become a critical task that system designers must face [1], [8]. Security protection of IASs emphasizes the concept of in-depth defense [9]. Security protection provided by field devices is the last defense of the in-depth defense; it provides intrinsic security for field control systems. Much research about security protection integrated control theory for field control systems has been presented [10], [11], and some literature has studied the security protection at the field control layer from the perspective of software implementation [12]–[14]. As the workflow of process industry automation is coherent, security protection of process industry automation should be done from a perspective of whole system. In addition to other security mechanisms such as industrial firewalls and intrusion detection and response are also two key elements of the dynamic security protection of IASs [6], [15], [16]. Research working in the field of security protection of IASs is focusing on intrusion detection technology [5], [17]–[20]. Comparatively, there are few studies about intrusion response of industrial automation [21]–[23].

Intrusion response systems (IRSs) can be classified into three types: 1) manual IRSs; 2) semiautomatic IRSs; and 3) automatic IRSs. IASs have high availability demands, which means the manual and the semi-automatic intrusion response will not meet the security protection requirements for IASs. Automatic intrusion response has been a research topic in the IT domain for several years. Foo et al. [24] presented an automated response mechanism in an intrusion tolerant system, called ADEPTS, which uses a graph of intrusion goals (I-GRAPH) as the underlying representation in the system. Zonouz et al. [21] proposed a new approach about automated intrusion response called RRE which can analyze undesired security events and their countermeasures based on attack-response trees. Fessi et al. [23] designed a multiattribute genetic algorithm (GA) approach for handling a multiattribute decision problem in intrusion response. These works focus on security policy decision and ignore the security policy execution, i.e., instant intrusion response.

In intrusion response, a security policy contains many different types of security services. Furthermore, in dynamic security protection, security policy changes in real-time [15].
For example, in a certain situation, a security policy generated by a security policy decision is composed of an encryption service and an access control service. When the attack situation changes, a security policy decision may generate a new security policy that is composed of access control services only. Each type of security service can be implemented by various realization tasks. For instance, the encryption service can be realized by many cryptographic algorithms, such as Rivest Shamir Adlemen (RSA), Rivest Cipher 4 (RC4), or data encrypt standard (DES). Therefore, the response task set generated from a security policy is also changing in real-time. On the other hand, system tasks of IASs have real-time characteristics. They are simple and fixed, and these tasks are generally scheduled by table-driven scheduling algorithms to achieve predictable behaviors [25], [26]. Thus, to execute a security policy in IASs, response tasks cannot be treated in a simple way (such as simply consider them as new, high priority tasks). Execution of the security policy without any countermeasures may affect the system task behaviors and the normal operation of the system [27]. For example, a security policy is composed of an encryption service that is realized by a task. If this task is executed without considering the execution of other tasks in the system, it may affect the real-time performance of these other tasks or make some tasks unable to be executed smoothly within the required time. The loss caused by inappropriate execution of a security policy may even be more than that caused by cyberattacks. IASs are quite different from IT systems in some ways. IASs have an inherent character of real-time response and determinacy that is embodied by real-time tasks and communication. In time-based security (TBS) theory, security is defined as a system that can detect and respond to cyberattacks before security threats happen or reach the target system [28], [29], i.e., timeliness is the key issue of online security protection.

Task scheduling is an important guarantee of effective task execution [30], [31]. There is some literature available to research task scheduling and optimization with multiconstraints [32]–[34]. Liu et al. [33] proposed a novel security constraint model based on the formulation of a scheduling problem for work-flow applications in distributed data-intensive computing environments. Saleh and Dong [34] designed a real-time multiagent design model and proposed an adaptive security-aware scheduling system that is combined with security service enhancement. These methods are designed for IT systems, and security requirements are regarded as constraints. They leave intrusion response tasks out of consideration. System tasks in IASs are generally simple and fixed, which are very different from the tasks in IT systems and are usually scheduled by table-driven scheduling algorithms [25], [26].

In this paper, we propose a general real-time control approach based on table-driven task scheduling of intrusion response in IASs. In this approach, the response tasks and system tasks are integrated into the task scheduling table with various requirements being considered to achieve instant intrusion response to cyberattacks. First, through a multiobjective optimization method based on the nondominated sorting GA II (NSGA-II), an intrusion response task set is generated from the security policy with consideration of security performance and cost in IASs. Second, a unified task model is conducted based on the combination of the intrusion response task set and a system task set. A distributed table-driven task scheduling and optimization scheme based on a GA and directed acyclic graph (DAG) is designed to update the task scheduling tables. Then, the system is reconfigured by updating the task scheduling tables to ensure the execution of intrusion response tasks and the normal operation of system functions. At last, the effectiveness of the proposed approach is verified by numerical simulations and application simulations based on a real-application system.

The rest of this paper is organized as follows. In Section II, the real-time issue of intrusion response in IASs is presented, and then a general real-time control architecture of intrusion response that contains response task set generation and integrated scheduling is proposed. Section III designs a multiobjective optimization method for response task set generation based on a given security policy. Integrated scheduling and optimization for the combination of the response task set and the system task set is presented by using a GA in Section IV. Sections V and VI show numerical simulations and application simulations, respectively. Finally, the study is concluded in Section VII with a summary of the research undertaken and plans for future research.

II. REAL-TIME CONTROL ISSUE OF INTRUSION RESPONSE FOR IASs

A. Problem Description

IASs are characterized by real-time and high reliability [35]. Meanwhile, IASs are industrial production systems and are safety critical systems. They should be able to defend themselves against the complicated and changing cyberattacks at runtime, i.e., they need to have dynamic security protection. The dynamic security protection for IASs is essential against cyberattacks [15]. Fig. 1 shows a general dynamic security protection model for IASs. Dynamic security protection of IASs consists of two parts: 1) intrusion detection and 2) intrusion response. Intrusion response is responsible for security policy decision and security policy enforcement. The security policy is generated at the security policy decision phase [22], while the security policy is interpreted and executed in the systems at the security policy enforcement phase.

Generally, a security policy is composed of different kinds of security services forming a security service
group (SSG) [36], and each type of security service can be implemented by various realization tasks. Thus, in the process of dynamic security protection, real-time security policies can be interpreted as different numbers and different types of response tasks that can take up system resources, such as communication bandwidth, CPU time, and memory. Therefore, the enforcement of these response tasks may disrupt the system task scheduling and affect the system’s normal operation. On the other hand, the response tasks generated from security policy directly determine the security performance of IRSs. Consequently, an integrated real-time task enforcement approach of intrusion response for IASs is needed to ensure the smooth and efficient enforcement of security policy and the system’s normal operation.

In this paper, we focus on the security policy enforcement component of intrusion response for IASs. A general real-time table-driven task scheduling approach of intrusion response is designed to solve this problem, where response tasks and system tasks are scheduled and optimized together with consideration of multiple constraints in IASs. The response tasks are generated dynamically according to a security policy, which are used to realize security services in the security policy. The system tasks include functional tasks and other nonfunctional (not pertaining to security) tasks.

**B. Formalizing the Problem**

An IAS is considered as a combination of a system task set $T^{sys}$ and a node set $N = \{p_i\}$. Suppose that the system tasks are dependent, nonpreemptible periodic tasks, and all nodes in the node set $N$ to be homogenous. IASs nodes include control devices, intelligent sensors, and intelligent actuators. Let $P$ represent a given security policy, and as mentioned above, the security policy can be regarded as an SSG composed of different kinds of security services [36]. So the security policy $P$ can be formalized as

$$P = \{S_1, \ldots, S_i, \ldots, S_k\} \quad (1)$$

where $S_i$ is a security service and can be realized by many realization tasks, that is

$$S_i = R_i = \{r_{i1}, r_{i2}, \ldots, r_{iq}\} \quad (2)$$

where $R_i$ is the realization task set of security service $S_i$. For example, the encryption service is a security service, which can be realized by many cryptographic algorithms, such as RSA, RC4, and DES. The parameters of a response task set $R_i$ are obtained offline. In this paper, we focus on security policy enforcement of IRSs based on a given security policy (shown as (1)), and do not discuss security policy decisions. Define $T^r$ is the response task set that is generated from security policy $P = \{S_1, \ldots, S_i, \ldots, S_k\}$, and the global task set $T$ is used to represent the combination of the response task set $T^r$ and the system task set $T^{sys}$. For easy of understanding, the notations and their meanings used in this paper are summarized in Table I.

![Fig. 2. Real-time control architecture of intrusion response for IASs.](image)

**C. Real-Time Control Architecture of Intrusion Response for IASs**

To ensure enforcement of the intrusion response of IASs, a real-time control approach of intrusion response based on table-driven task scheduling and optimization is proposed, which is shown in Fig. 2.

As shown in Fig. 2, the intrusion detection and the security policy decision components are executed periodically, and the security policy generated from the security policy decision component is the real-time input of security policy enforcement component. This is a two-stage process containing response task set generation and integrated scheduling. In the response task set generation stage, with consideration of security performance requirements and time constraints, the Pareto solution of a response task set is generated from the security policy through the NSGA-II. Then, a solution in the form of a response task set having the highest security
performance in the Pareto solution is selected as the optimal response task set. In the integrated scheduling stage, the response task set $T^*$ is first combined with system task set $T_{sys}$ to generate the global system task set $T$. Then, based on a DAG, a GA-based task scheduler is used to reconfigure the global system scheduling tables. If the global system task set is unschedulable, the response task set with slightly lower security performance will be re-elected as the optimizing response task set. The flowchart of the task scheduling and optimization is shown in Fig. 3.

III. RESPONSE TASK SET GENERATION OF INTRUSION RESPONSE FOR IASs

In this section, a response task set generation scheme of intrusion response is presented. We first model the problems of response task set generation, then we solve this problem using NSGA-II.

A. Modeling the Problem

The realization task $r_{ij}$ is defined as $r_{ij} = (c_{ij}, w_{ij}, cc_{ij}, pt_{ij})$, where $c_{ij}$ and $w_{ij}$ are the cycle time and (worst case) execution time of the realization task $r_{ij}$, respectively. $cc_{ij}$ is defined as the communication load of the realization task $r_{ij}$, which is represented as the byte count of the exchanged data of the realization task $r_{ij}$ in one period time $c_{ij}$. $pt_{ij}$ is the security protection time that the realization task $r_{ij}$ can provide, which means the duration time that the attacker is able to achieve an attack goal in the system when the realization task $r_{ij}$ is employed [28], [29]. Thus, $pt_{ij}$ of the $r_{ij}$ is different for the different types of attacks. The first three parameters of the $r_{ij}$ can be determined in advance through evaluation [31], [36], [37], while the $pt_{ij}$ of the $r_{ij}$ for each type of attack can be listed online and determined in a security policy decision. It is important to note that heterogeneous processors are not considered here, i.e., all processors are treated the same.

Then, the response task set $T^* = \{t^*_1, \ldots, t^*_i, \ldots, t^*_n\}$ is generated from the security policy $P = \{S_1, \ldots, S_i, \ldots, S_k\}$. Each individual response task $t^*_i$ is chosen from the corresponding realization task set $R_i$. Thus, a security policy may have different security performances because a security policy can be realized by different SSGs that may have different security performances. For instance, the security performance of two cryptographic algorithms (e.g., RSA and DES) is obviously different. Security performance can be measured by time, i.e., TBS theory [28]. Security is defined as an intrusion attack that is detected and responded to in time [28], [29]. Thus, systems must be recovered before attackers achieve their intrusion goals. As shown in Fig. 4, security is formalized as follows:

$$PT > Det_t + Res_t$$

where $PT$ is the security protection time provided by the SSG, $Det_t$ and $Res_t$ are the intrusion detection time and the intrusion response time, respectively. The latter two parameters can be considered as constants for a specific system. Security time margin (STM) is defined as the difference between $PT$ and $Det_t + Res_t$, i.e., $STM = PT - (Det_t + Res_t)$. Thus, $STM \leq 0$ represents a system that is insecure, while $STM > 0$ represents a system that is secure. The STM can represent the security level.

For competition reasons, many intelligent devices at field control layer in IASs are implemented based on embedded systems, where the CPU performance, communication bandwidth, and memory size of IASs are designed according to rather limited requirements. Because of the business competition in IASs, cost is an important factor that must be considered during design and implementation. Most intelligent devices at the field control layer in IASs are implemented based on embedded systems, where the CPU performance, communication bandwidth, and memory size are designed according to rather limited requirements. Thus, in this paper, the IRS is constructed under certain resource constraints. In addition, the IRS is used to mitigate the effect of a cyberattack and protect system functions, which are implemented in IAS nodes. Thus, IAS nodes share the CPU resources and communication bus with the IRS, which requires that the response tasks of the IRS always affects the system’s normal operation. So, the response tasks of IRS should be considered together with system function tasks to mitigate impacts of the IRS on system functions.

Therefore, three optimization goals should be considered in the process of the response task set generation: security performance, the communication load and execution time of task. Let a decision variable $x_{ij}$ denote whether the $j$th realization task in realization task set $R_i$ is selected to implement the
ith security service $S_i$, and $X = (X_1, \ldots, X_i, \ldots, X_k)$, where $X_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ik})$

$$x_{ij} = \begin{cases} 1, & r_{ij} \text{ is selected} \\ 0, & r_{ij} \text{ is not selected} \end{cases}$$  

(4)

$x_{ij}$ should satisfy the following constraint: $\sum_{j=1}^{q_i} x_{ij} = 1$.

Then, the security protection time of the response task set $T'_i = \{t'_1, \ldots, t'_i, \ldots, t'_k\}$ can be calculated as

$$PT = \sum_{i=1}^{k} \sum_{j=1}^{q_i} x_{ij}p_{ij}.$$  

The communication load per unit of time of the response task $t_{ij}$ is $c_{ij}$ and the communication load of the response task set can be represented as $CC = \sum_{i=1}^{k} \sum_{j=1}^{q_i} (x_{ij}c_{ij}/c_{ij})$. Analogously, the sum of the execution times of the response tasks in the response task set can be represented as $ET = \sum_{i=1}^{k} \sum_{j=1}^{q_i} (x_{ij}w_{ij}/c_{ij})$. Meanwhile, as shown in Fig. 4, to ensure that the system is secure, the following constraint must be satisfied: $PT \geq (Det_i + Res_t)$, with $Det_i$ and $Res_t$ considered as constants for a specific system.

To generate the response task set $T'_i$ the problem is formalized as a multiobjective optimization problem (MOOP). The objective functions of this MOOP model are stated as

$$\begin{align*}
\text{Min } ET &= \sum_{i=1}^{k} \sum_{j=1}^{q_i} x_{ij}w_{ij}/c_{ij} \\
\text{Min } CC &= \sum_{i=1}^{k} \sum_{j=1}^{q_i} x_{ij}c_{ij}/c_{ij} \\
\text{Max } PT &= \sum_{i=1}^{k} \sum_{j=1}^{q_i} x_{ij}p_{ij}.
\end{align*}$$

(5)

And the constraints of the MOOP model are expressed as

$$\begin{align*}
PT &> (Det_t + Res_t) \quad (6a) \\
\sum_{j=1}^{q_i} x_{ij} &= 1. \quad (6b)
\end{align*}$$

(6)

Meanwhile, the decision variable is represented as

$$X = (X_1, \ldots, X_i, \ldots, X_k)$$

(7)

where $X_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{ik})$. Then, the problem of response task set generation can be formalized as (5) and (6).

**B. Using the NSGA-II to Solve the MOOP Problem**

Many evolutionary algorithms have been developed to solve MOOP problems, such as simulated annealing algorithm, particle swarm optimization algorithm, ant colony optimization algorithm, NSGA, and NSGA-II [38]–[43]. NSGA is proposed by Srinivas and Deb [39] to deal with MOOPs, where Goldberg’s nondomination criterion is used to determine solution ranks and fitness sharing is used to control the diversity of solutions in the search space. To improve the high-level sensitivity of NSGA to fitness sharing parameters, Deb et al. [44] introduced NSGA-II which includes a second-order sorting criterion called crowding distance and is faster and more reliable than NSGA. NSGA-II has been demonstrated to be among the most efficient algorithms for multiobjective optimization on a number of benchmark problems [44], as it has excellent search capability and flexibility [45]. Thus, we selected NSGA-II to solve the application problem of response task generation here.

NSGA II has good convergence toward the Pareto frontier without losing the diversity of the solutions, which introduces elitism into an evolutionary process and guarantees a diversity preserving mechanism. At the start of NSGA II, an initial population $P_0$ is randomly generated. Then, objective function values of the population $P_0$ are evaluated and the two criteria of each of the individual $i$‘s in $P_0$, rank $r_{i\text{rank}}$ and crowding distance $i_{\text{distance}}$, are calculated. After the initialization, the program enters a while loop. In the loop, there are six steps.

1) Selecting parent individuals from $P_t$ based on tournament selection strategy.
2) Generating offspring population $Q_t$ by crossover and mutation operators.
3) Evaluating objective function values of the population $Q_t$.
4) Merging $P_t$ and $Q_t$ to create $R_t$.
5) Calculating the two criteria $(r_{i\text{rank}}, i_{\text{distance}})$ for each of the individual $i$ in $R_t$.
6) Selecting the best individuals in $R_t$ to generate $P_{t+1}$.

These steps are repeated until the stopping condition is met. In the above steps, $P_t$ is the original population, $Q_t$ is the newly created population according to $P_t$, and $R_t$ is the combination of the populations of $P_t$ and $Q_t$. There are sophisticated algorithms to calculate the domination rank $r_{i\text{rank}}$ and the crowding distance $i_{\text{distance}}$, such as described in [46], so that will not be covered again here.

Then, according to the MOOP problem shown as (5)–(7), the chromosome structure, genetic operations and optimal solution selection from the Pareto solution are designed as follows.

1) **Chromosome Structure:** The chromosome structure is determined by the decision variable in the problem. According to (7), each individual is coded as a binary vector $X$ of dimension $\sum_{i=1}^{k} q_i$ naturally, representing a decision vector.

2) **Genetic Operations:** Genetic operations are used to produce a new generation from an old generation. To effectively reduce the amount of the potential solutions, the crossover operator and the mutation operator are designed to meet the constraint shown as (6b).

1) **Crossover Operator:** In this research, the arithmetic crossover operator is used to produce offspring by swapping genetic parts of the parents. Thus, crossover point selection is key for the crossover operator. According to (6b), each $X_i$ should be regarded as an integral gene segment that cannot be divided. Therefore, the set of alternative crossover points is $\{q_1, q_1 + q_2, \ldots, \sum_{j=1}^{i-1} q_j, \ldots, \sum_{j=1}^{i-1} q_j\}$. The crossover point is randomly selected in the set, and the corresponding gene segments are exchanged to generate the offspring.

2) **Mutation Operator:** In the mutation operator process, a single bit in the parent string is flipped to create a new offspring string. However, to meet the constraint shown as (6b), first, a single bit randomly selected in the parent string is flipped, resulting in two cases.

a) **Case 1:** If the selected bit is “0,” then in the corresponding gene segment $(X_i)$, the bit which...
Fig. 5. Architecture of the integrated scheduling component in Fig. 2.

is “1” is changed to 0, and the selected bit is changed to 1.

b) Case 2: If the selected bit is 1, then the selected bit is changed to 0, and a bit randomly selected in the residual bits of the corresponding gene segment \( X_i \) is changed to 1.

3) Optimal Solution: The optimal solution for a response task set should be selected from the Pareto solution. In the process of multiobjective optimization, there are three optimization objectives: 1) security protection time; 2) communication load; and 3) execution time. In the Pareto solution, we select the optimal solution by making security protection time the highest priority. First, the solution with the largest security protection time in the Pareto solution is reselected as the optimal solution. The following section introduces the task schedule for the combination of the tasks.

IV. INTEGRATED SCHEDULING OF INTRUSION RESPONSE FOR IASs

Response task set \( T^r \) will be executed with system task set \( T^3\text{s} \) after it is generated. In this section, an integrated scheduling scheme based on a DAG task model and GA is proposed for IASs. The architecture of the integrated scheduling is shown in Fig. 5. In the scheme initially, a global task set is first generated through unified modeling of the response tasks and system tasks. Then, global tasks are mapped and scheduled based on a DAG task model and GA, to achieve a scheduling table of each node of the system. During this process, communication tasks in nodes can be regarded as system tasks. Therefore, the communication scheduling table is also determined. Finally, the system is reconfigured by updating the scheduling tables to protect against cyberattacks.

A. System Model

As mentioned in Section II-B, an IAS is considered as a system task set \( T^3\text{s} = \{\tau_i\} \) and a node set \( N = \{pl_i\}; \ i = 1, \ldots, m \). The global task set \( T = \{\tau_i\}; \ i = 1, \ldots, n \) is c combination of the system task set \( T^3\text{s} \) and the response task set \( T^r \), i.e., \( T = T^r \cup T^3\text{s} \), where \( T^r = \{\tau_i^r, \ldots, \tau_i^3\} \) and \( \tau_i^r = (c_i, w_i, t_i, pl_i) \).

Task \( \tau_i \) in the global task set is identified by the triples \( \tau_i = (w_i, st_i, pl_i) \), where \( w_i \) is the worst execution time of the task, \( st_i \) is the start time of the task (relative to the start time of the macro cycle), and \( pl_i \) is a list of the available node where the task can be assigned. Meanwhile, to model the global task set, a DAG is used to describe the dependent relationships among the global tasks. The global task set is represented by a binary group \( G = (T, E) \), where \( T \) is the global task set, and \( E = \{ei_j[i, j = 1, \ldots, n] \} \) is the set of directed arcs or edges between the tasks to represent the dependency. The \( \omega(ei_j) \) represents the communication load (data exchanged from \( \tau_i \) to \( \tau_j \)) per one cycle time from \( \tau_i \) to \( \tau_j \). The edge \( ei_j \in E \) represents the dependent relationship that task \( \tau_i \) cannot start the execution of until task \( \tau_j \) completes its execution. Fig. 6 shows an example of the task model presented by DAG.

To facilitate constructing the DAG task model, the macro cycle time (MCT) is defined as the least common multiple of the cycle times of all the tasks in \( T \) and then all tasks can be converted into the tasks with the same cycle time. For example, assuming \( T^r \cup T^3\text{s} \) be \( \{\tau_1, \tau_2, \tau_3\} \) and the cycle times of these tasks be 2, 4, and 4 ms, respectively, the MCT equals 4 ms, and \( \tau_1 \) can be converted into two tasks whose cycle times are 4 ms. So, the parameter \( c_i \) in \( \tau_i^r \) can be ignored. The parameter \( w_i \) in \( \tau_i^r \) is the same as \( w_i \) in \( \tau_i \). The parameters \( cc_i \) and \( pl_i \) are not used in the integrated scheduling, so they can also be ignored. As the security services are usually not coupled with any hardware, the \( pl_i \) of \( \tau_i^r \) equals \( N \).

Define the concept of the prodromal task set \( \text{Pred}(\tau_i) \) and successive task set \( \text{Succ}(\tau_i) \) as follows:

\[
\text{Pred}(\tau_i) = \Gamma, \quad \text{If } ei_j \in E, \quad \tau_j \in \Gamma
\]
\[
\text{Succ}(\tau_i) = \Phi, \quad \text{If } ei_j \in E, \quad \tau_j \in \Phi
\]

If the prodromal task set is a null set, the corresponding task is called a start task; if the successive task set is null set, the corresponding task is called an end task. As shown in Fig. 6, \( \tau_1 \) and \( \tau_2 \) are the start tasks, while \( \tau_3 \) and \( \tau_9 \) are the end tasks. The prodromal task set of \( \tau_4 \) is \( \{\tau_2, \tau_3\} \) and the successive task set of \( \tau_4 \) is \( \{\tau_5, \tau_6\} \).

B. Task Mapping and Scheduling

As mentioned in Section IV-B, the global tasks are described by triples \( \tau_i = (w_i, st_i, pl_i) \) and the DAG model \( G \). In the process of the integrated task mapping scheduling, the following constraints should be taken into account.

1) Coupling Relationship Between Tasks: For a specific application in IASs, some tasks might need to run in the same node. Besides, as described in Section IV-A, the tasks which are converted from a same task must run
in the same node. The coupling relationship between $\tau_i$ and $\tau_j$ can be represented as: if $\tau_i$ runs in $p_k$, $\tau_j$ also runs in $p_k$. In order to facilitate subsequent processing, we assume the coupled tasks are adjacent to each other.

2) **Dependent Relationship Between Tasks:** The dependent relationship between tasks is described by DAG, which can be expressed as: if $e_{i,j} \in E$, $s_j \geq s_i + w_i$.

3) **Dependent Relationship Between Tasks and Nodes:** For a specific application in IASs, some tasks can only be executed by specific nodes. For instance, a data acquisition task must run in the corresponding sensor node. In task $\tau_i = (w_i, s_i, pl_i)$, $pl_i$ lists the nodes in which the task $\tau_i$ can run.

4) **Communication Cost Constraint:** The weight of the edge in DAG $w(e_{i,j})$ is used to express the communication load from $\tau_i$ to $\tau_j$. In IASs, the communication bandwidth is limited, thus, we consider the communication load of the system being the sum of the communication load between nodes.

5) **Load Constraint:** In the process of task mapping, at least one task should be assigned to each node.

IASs belong to distributed systems, and communication load between nodes is an important index of the task mapping and scheduling in distributed systems [47]. In our research, we select the communication load as the optimization goal, and formulate the task mapping and scheduling as an optimization problem.

We used binary decision variable with two indices: $Y = y_{i,j}; i = 1, 2, \ldots, n; j = 1, 2, \ldots m$ to denote the tasks mapping on nodes, where $y_{i,j}$ is defined as follows:

$$y_{i,j} = \begin{cases} 1, & \text{if } \tau_i \text{ is mapped on } p_j \\ 0, & \text{otherwise}. \end{cases} \quad (9)$$

Because one task can only be mapped to one node, the $y_{i,j}$ must satisfy the following constraint:

$$\sum_{j=1}^{m} y_{i,j} = 1. \quad (10)$$

Define a function $Q(i,j)$ to denote whether two tasks are mapped on the same node

$$Q(i,j) = \begin{cases} 1; & \text{if } \tau_i \text{ and } \tau_j \text{ are mapped on the same node} \\ 0; & \text{otherwise}. \end{cases} \quad (11)$$

And then, the communication load of the system can be calculated as follows:

$$\text{Comm} = \sum_{e_{i,j} \in E} Q(i,j) \times w(e_{i,j}). \quad (12)$$

To solve the task mapping and scheduling problem by a GA, the binary decision variables $Y = \{y_{i,j}\}; i = 1, 2, \ldots, n; j = 1, 2, \ldots m$ are used as the chromosome. The constraints 1) and 3) are satisfied by crossover and mutation operations, and the constraints 2), 4), and 5) are used to delete illegal individuals in the offspring population.

1) **Crossover Operation:** In the process of cross, $Y_i = [y_{i,1}, \ldots, y_{i,j}, \ldots, y_{i,m}]$ is regarded as a meta-segment to meet (10). Then meta-segments are recombined through the single-point crossover to generate offspring, with the available crossover point set being $\{1, 2, \ldots, n\}$. To meet constraint 1), numbers of the coupled tasks are removed in the available crossover point set. For example, if $\tau_2$ and $\tau_4$ are coupled, then, 2 and 3 are removed in the available crossover point set. Obviously, if the parents meet constraint 3), the offspring also meets constraint 3).

2) **Mutation Operation:** To meet (10), operation is similar to the mutation operation in Section III-B. To meet constraint 1), if $\tau_i$ and $\tau_j$ are coupled and $Y_i$ mutate, then, let $Y_j$ equal $Y_i$ in each offspring.

After offspring population is generated by crossover and mutation operations, it should be checked by constraints 2), 4), and 5). For constraint 2), the dependent relationship between tasks is represented by DAG. Obviously, the completion of all the prodromal tasks of task $\tau_i$ is a necessary condition for task $\tau_i$ to be started. Thus, tasks in the DAG are layered according to the dependent relationship in Algorithm 1. task_level is the number of task layers, and $F_i$ is the task set to which the task belongs to the $i$th layer. Considering the whole DAG, the sum execution time (SET) that all $n$ tasks execute on $m$ nodes must not exceed the MCT. Then, for each individual represented by $Y$, the SET can be calculated based on the layered tasks in Algorithm 2, and the start times ST of all tasks are also determined together. The $m(t)$ is used to record the available time of the $i$th node. In addition, it is easy to check each individual for constraints 4) and 5). After the optimal solution is obtained, the decision
Algorithm 2 Task Scheduling Algorithm in Integrated Scheduling

Require: Y F = {F1, F2, ..., F_task_level}
Ensure: ST = {s1, s2, ..., s_l} SET
1: [nt(1), nt(2), ..., nt(m)] = [0, 0, ..., 0]
2: for i = 1 to task_level - 1 do
3: M_i = N_i = {}
4: for all τ such that τ ∈ F_i do
5: if Succ(τ) = NULL then
6: N_i.append(τ)
7: else
8: M_i.append(τ)
9: end if
10: end for
11: sort(M_i(τ, w))
12: sort(N_i(τ, w))
13: for j = 1 to len(M_i) do
14: τ = M_i(τ_j)
15: τ.st = Max_τj∈Pred(τ) (nt(τ_j) + τ_k + w_k)
16: nt(τ(τ)) = τ.st + τ.w
17: end for
18: for j = 1 to len(N_i) do
19: τ = N_i(τ_j)
20: τ.st = Max_τj∈Pred(τ) (nt(τ_j) + τ_k + w_k)
21: nt(τ(τ)) = τ.st + τ.w
22: end for
23: end for
24: N = N_i
25: SET = Max_{τ ∈ N} (τ_st + τ_k + w_k)
26: return SET ST

variable Y and the start time vector of all tasks ST are determined, i.e., the new task schedule tables are generated. Then, global task set T solution is not found, as shown in Fig. 2, the response task set T’ will be reselected, which is described in Section III-B.

V. NUMERICAL SIMULATIONS

In this section, the proposed method is implemented based on C++, and these simulations are carried out on a 2.6-GHz Pentium 4 laptop with 2 GB RAM to analyze and evaluate the performance and efficiency of the proposed real-time intrusion response approach in IASs. Four numerical scenarios are shown in Table II. As described in Section II-A, the number of response tasks k represents the dimension of the security policy. Table III shows the parameters predetermined of the corresponding realization task sets for the simulation. We suppose that the security policies are PScenario1 = {S1, S2, S3, S4, S5}, PScenario2 = {S1, S2, S3, S4, S5}, PScenario3 = {S1, S2, S3, S4, S5}, and PScenario4 = {S1, S2, S3, S4, S5} for the four scenarios, respectively. Table IV shows the parameters of the system task set for the simulation. In scenario 1, the system task set is {τ1, τ2, τ3, τ4, τ5, τ6}; in scenario 2, the system task set is {τ1, τ2, τ3, τ4, τ5}; in scenario 3, the system task set is {τ1, τ2, τ3, τ5, τ6, τ7, τ8, τ9, τ10}; and in scenario 4, all tasks are used and tasks τ1–τ10 are used twice. The DAGs are randomly generated based on the corresponding task sets.

A. Analysis of the Response Tasks Generation

During the process of the response tasks generation, the Pareto solution of the response task set is generated through NSGA II, and then the optimal solution of the response task set is selected according to the principle described in Section III-B. There are two main performance metrics for response task generation based on multiobjective optimization: 1) convergence to the Pareto optimal front and 2) the maintenance of a diverse set of solutions. Fig. 7(a) shows the Pareto solution fronts of response task generation in scenarios 1 and 2 at the 50th simulation step, and the relations between each pair of the three optimization objectives of the Pareto optimal are presented in Fig. 7(b)–(d). They show the tradeoff among security protection time, sum of communication loads of the response tasks, and sum of execution times of the response tasks. After selecting the optimal solution in the Pareto solutions, the result for the response task set generation in scenario 1 is T*Scenario1 = {r11, r22, r32, r43}, and the results for the response task set generation in scenario 2 is T*Scenario2 = {r11, r22, r32, r43, r52}. The optimization solutions of the response task set generation are associated with the given security policy and the realization task sets, but not with the system task set and system nodes. Thus, Pareto optimal fronts of scenarios 2–4 are the same. Based on the above results, it can be seen that the Pareto solution of the response tasks generation are convergent and the Pareto solution fronts uniformly scattered.

B. Analysis of the Integrated Scheduling

The global system tasks (including response tasks and system tasks) are mapped and scheduled synthetically through GA based on the DAG model. Fig. 8 shows the variation of the fitness (communication load) for four scenarios during the genetic computation. It is found that all the individuals have reached almost the same fitness value after about ten generations for the four scenarios, which indicates that GA has reached the optimal solution after about ten generations. In Fig. 8, we can infer that the appropriate generation number of GA in the integrated scheduling module depends not only on the number of tasks but also on the number of the nodes in the system. This is mainly because the number of gene bits is determined by the product of the task number and the node number.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>n</th>
<th>m</th>
<th>k</th>
<th>MCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>1000</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>1000</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>6</td>
<td>5</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>8</td>
<td>5</td>
<td>1000</td>
</tr>
</tbody>
</table>
Table IV shows the results of the task mapping and scheduling for the four scenarios when the related parameters are set as follows: initial_population = 100 and num_generation = 20. The results show that all tasks can be completed in one MCT, which means that all tasks are schedulable satisfying all constraints, and the execution of the response tasks has no effect on the system tasks. This demonstrates that the proposed method can achieve instant intrusion response.

C. Real-Time Performance Evaluation

The proposed approach aims to guarantee the smooth online enforcement of a security policy. On the other hand, the intrusion response that includes the proposed approach is also implemented as a task, so the execution time needs to be evaluated. Thus, we must consider the real-time performance of the proposed approach. To evaluate the real-time performance, 130 repeated tests for the four scenarios are carried out. Fig. 9 shows the real-time performance of the proposed approach of intrusion response based on task scheduling and optimization for the four scenarios in the repeated tests. It can be seen that the execution times of the proposed method for all four scenarios are less than 60 ms. The mean values are 14.82, 26.50, 26.58, and 46.28 ms, respectively. This is because the number of the gene bits in the response task set generation based on NSGA is \( \sum_{k=1}^{m} q_k \) and the number of the gene bits in the integrated scheduling based on GA is \( m \times n \). These two parameters are the important factors influencing the execution times of the proposed method. These results show that the proposed algorithm can be executed in a short time. It also means that a system can respond quickly to any cyber attacks.

VI. SIMULATION—SIMPLIFIED TEP CONTROL SYSTEM

To further verify the effects and feasibility of the proposed approach of intrusion response in real-application systems, in this section, we construct a simulation environment based on OPEN and MATLAB, and a simplified Tennessee Eastman process (TEP) control system is used to carry out the simulations.

A. Simulation Setup

1) Simplified TEP and Simulation System: We used a simplified TEP control system as Ricker [48] described to carry out the simulations. The simplified TEP control system is a chemical reactor plant and has been previously used in [49] and [50]. In the chemical process, two reactants, A and C, are used to generate the product D, while B is an inert product. The schematic of the simplified TEP control system is shown in Fig. 10. More details on the simplified TEP can be found in [49] and [50].

In the simulation, we focused on a field control system for industrial automation and do not consider links to the outside networks. The simplified TEP model and the associated multiloop PI control law proposed in [48] are used. The communication network is a controller area network (CAN) bus, which is also implemented in the OPNET environment. The model of the physical process is called an agent node (PA in Fig. 11) in OPNET. The simulation system contains one master node (MN) and five slave nodes (one plant agent node and four controller nodes). All nodes are connected to a CAN bus to exchange messages.

Based on the above simulation system, the dynamic security protection shown in Fig. 1 is implemented. It is important to note that this paper focuses on security policy enforcement, thus the rest of the dynamic security protection is only realized in a simple way. An intrusion detection module is used...
to generate a security policy online that is used for the input to the security policy enforcement. The intrusion detection module can generate an integer that ranges between 0 and 3. 0 represents no cyberattack, while “1,” “2,” and “3” represent three types of cyberattacks. The intrusion detection module is implemented in all the nodes and used to simulate the intrusion detection component of Fig. 1. The intrusion response module contains two parts: 1) security policy decision and 2) security policy enforcement, which is implemented in the MN. To implement the security policy decision, each type of attack corresponds to a security policy \( P \). The parameters of the security policies are predetermined, and are presented in Table VI. The proposed method of security policy enforcement is also implemented according to the descriptions in Sections III and IV. The process models of the MN node and the other nodes are presented in Fig. 12. In the MN, the DETECT process contains one task, i.e., intrusion detection, and the RESPONSE process also contains only one task, i.e., intrusion response. The CAN_TX and CAN_RX processes are responsible for the network communication for the node. The other tasks are in the APP process. Because we only consider the execution time of intrusion detection in this paper, in other nodes (slave nodes), a useless task is used to simulate the real-intrusion detection in the DETECT process. The APP, CAN_TX, and CAN_RX processes are these processes in the MN. As shown in Fig. 2, the updated scheduling tables are broadcasted through the CAN bus after they are changed. All nodes are reconfigured according to the corresponding scheduling tables.

2) Parameter Configuration: Table VI shows the system tasks of the simplified TEP control system. The execution times of these tasks are obtained by estimation, according to the corresponding code. The periods \( c_i \) and the \( pl_i \) are determined by the application. In the intrusion detection task, some dummy code is inserted to simulate the execution time of the real-intrusion detection. In practice, the PA node is nonexistent, as it is only for communication between the physical system and the control system. Thus, the PA node and its tasks are not considered in the following simulations. To facilitate understanding, we label the nodes MN, FC, PC1, PC2, and CC for the simulation system shown in Fig. 11 as \( p_1-p_5 \), respectively.
The decision table for the security policy is designed as follows: when there is no attack, the security policy $P$ is an empty set; when an attack of Type 1 occurs, the security policy is $P_{Type1} = \{S_1, S_2\}$; when an attack of Type 2 occurs, the security policy is $P_{Type2} = \{S_1, S_3\}$; for a Type 3 attack, the security policy is $P_{Type3} = \{S_1, S_2, S_3\}$. This means that the response tasks are generated from the corresponding security policy $P$. The security services $S_1$–$S_3$ that are used in the security policy can be implemented by the corresponding realization task sets $R_1$–$R_3$, respectively. The parameters of the realization tasks are given in Table VII. These realization tasks are prestored as functions in the nodes.

B. Simulation Results

To verify the effectiveness of the proposed method in the simulation system, the intrusion detection component generates the following output:

$$\text{Output} = \begin{cases} 0, & t \in [0, 10) \cup [70, 100) \\ 1, & t \in [10, 30) \\ 2, & t \in [30, 50) \\ 3, & t \in [50, 70) \end{cases} \quad (13)$$

Table VIII shows the Pareto solution for the response task set generation for $t = 50$. As described in Section III-B, the
optimal response task set is the one with the largest security protection time in the Pareto solution. Thus, the optimal solution of the response task set generation is $T^* = \{r_{11}, r_{23}, r_{32}\}$.

Then, through the integrated scheduling module, we can obtain the optimal solution of the integrated task mapping and the corresponding task scheduling table. Due to limited space, Table IX only shows the integrated task mapping and the corresponding task scheduling table for tasks reconfigured for $t = 50$. The result shows that all tasks (including system tasks and response tasks) are schedulable, and the sum of the execution times of all the nodes does not exceed the MCT (3000 ms). It also means that response tasks can successfully be carried out with no effect on the execution of the system tasks.

Table IX shows the execution of all the tasks from the perspective of task scheduling. Control performance is also an important factor for IASs. From the perspective of control engineering, it should also be guaranteed that the execution of a security policy will have no effect on the control performance of the system. Fig. 13 provides a comparison of dynamic response $y_F$ between the normal simplified TEP control system and the system with dynamic security protection. It clearly shows that the dynamic security protection has almost no influence on the dynamic response of the simplified TEP control system.
In the above simulations, there are three different security policies. This is because the proposed approach of intrusion response can be clearly explained based on these configurations, and security policy decision is not the focus of this paper. In real-time security protection of IASs, security policies generated from the security policy decision is complex and varied. From the above simulation results, it can be seen that the proposed real-time control approach of intrusion response is an effective method to guarantee the smooth, timely execution of the security policy without effect on system control performance.

VII. Conclusion

Timely intrusion detection and response is the key idea for online security protection for IASs. In this paper, we propose a general real-time control approach based on table-driven task scheduling of intrusion response for IASs, which utilizes TBS criteria and integrates the response tasks and system tasks into the scheduling and optimization after consideration of a number of requirements. First, to meet the wide variety of security requirements for security protection of IASs, the security policy is formalized as a group of security services with different types. Each type of security service can be realized by many tasks, and these compose the realization task set for the corresponding security service. The response tasks are generated through an NSGA-II algorithm according to the security policy. Then, to mitigate the execution of the response tasks, an integrated scheduling strategy based on the GA is designed to map and schedule the system tasks and response tasks together.

The proposed method has been implemented using the C++ language and analyzed on several synthetic applications. Then, a simplified TEP control system was used to further verify the effectiveness of the proposed method in dynamic security protection based on OPNET and MATLAB environments. Simulation results have shown that the given security policy can be smoothly implemented and achieve the instant intrusion response without effect on the normal operation of the system.

Currently, all nodes are considered as homogeneous. In fact, there are also many heterogeneous nodes in most IASs. Further improvement to this paper can be made by considering the inclusion of heterogeneous node. In addition, this paper focused on security policy enforcement, and did not discuss security policy generation. Therefore, security policy generation based on decision theory of intrusion response will be considered.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their helpful comments and suggestions for improving the quality of this paper.

REFERENCES


Naixue Xiong (M'08–SM'12) received the Ph.D. degrees in software engineering and dependable networks from Wuhan University, Wuhan, China, and the Japan Advanced Institute of Science and Technology, Nomi, Japan. He was with the Wentworth Technology Institution, Georgia State University, Atlanta, GA, USA, for several years. He is currently a Full Professor with the School of Computer Science, Colorado Technical University, Colorado Springs, CO, USA. His current research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory.

Prof. Xiong was the General Chair, the Program Chair, the Publicity Chair, a PC Member, and an OC Member of over 100 international conferences, and a Reviewer of about 100 international journals, including the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON MOBILE COMPUTING, and the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. He serves as the Editor-in-Chief and an Associate Editor or an Editor Member for over ten international journals, including an Associate Editor for the IEEE TRANSACTION ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, the Editor-in-Chief for the Journal of Parallel and Cloud Computing, and a Guest Editor for over ten international journals, including the Sensor Journal, WINET, and MONET.

Shuang-Hua Yang (M'05–SM'06) received the B.S. degree in instrument and automation and the M.S. degree in process control from the University of Petroleum, Shandong, China, in 1983 and 1986, respectively, and the Ph.D. degree in intelligent systems from Zhejiang University, Hangzhou, China, in 1991.

He is currently a Professor of Networks and Control and the Director of the Networks, Communications and Control Systems Research Division, Loughborough University, Leicestershire, U.K. His current research interests include wireless sensor networks, networked control, safety critical-systems, and real-time software maintenance.

Prof. Yang is an Associate Editor of the International Journal of Systems Science, the International Journal of Automation and Computing, and the Arabian Journal for Science and Technology. He is an Editorial Advisory Board Member of the International Journal of Information and Computer Security and the Journal of Measurement and Control. He is a Fellow of the Institute of Measurement and Control and a Chartered Engineer in the U.K.

Yuanqing Qin received the M.S. and Ph.D. degrees in control theory and control engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2003 and 2007, respectively.

He is currently a Lecturer with the Department of Control Science and Engineering, Huazhong University of Science and Technology. His current research interests include networked control systems, artificial intelligence, and machine vision.

Qi Zhang was born in 1987 in Liaoning, China. He received the B.S. degree in automation from the Huazhong University of Science and Technology, Wuhan, China, in 2011, where he is currently pursuing the Ph.D. degree in control theory and control engineering with the School of Automation under the supervision of C. Zhou.

His current research interests include distributed systems/networks and industrial Ethernet.